

Rexroth IndraLogic XLC IndraMotion MLC 12VRS HMI Connection

R911334184
Edition 01

Application Manual



Title	Rexroth IndraLogic XLC IndraMotion MLC 12VRS HMI Connection		
Type of Documentation	Application Manual		
Document Typecode	DOK-XLCMLC-HMI*****V12-AP01-EN-P		
Internal File Reference	RS-8962c7c218e804620a6846a00139fd6a-1-en-US-8		
Purpose of Documentation	This documentation describes the visualization systems supported by IndraLogic XLC and IndraMotion MLC/MLP and their connection.		
Record of Revision	Edition	Release Date	Notes
	Edition 01	11.2011	First edition
Copyright	© Bosch Rexroth AG 2011 This document, as well as the data, specifications and other information set forth in it, are the exclusive property of Bosch Rexroth AG. It may not be reproduced or given to third parties without its consent.		
Liability	The specified data is intended for product description purposes only and shall not be deemed to be a guaranteed characteristic unless expressly stipulated in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.		
Published by	Bosch Rexroth AG Bgm.-Dr.-Nebel-Str. 2 • 97816 Lohr a. Main, Germany Phone +49 (0)93 52/40-0 • Fax +49 (0)93 52/40-48 85 http://www.boschrexroth.com/ System Development Automation Motion Logic Control, TR (FrWe/PiGe)		
Note	This document has been printed on chlorine-free bleached paper.		

Table of Contents

	Page
1 About this Documentation.....	7
1.1 Validity of the Documentation.....	7
1.2 Required and Supplementing Documentations.....	7
1.3 Information Representation.....	11
1.3.1 Safety Instructions.....	11
1.3.2 Symbols Used.....	11
1.3.3 Names and Abbreviations.....	12
2 Important Instructions for Use.....	13
2.1 Intended Use.....	13
2.1.1 Introduction.....	13
2.1.2 Scope of Application.....	13
2.2 Improper Use.....	14
3 Safety Instructions for Electric Drives and Controls.....	15
3.1 Definitions of Terms.....	15
3.2 General Information.....	16
3.2.1 Using the Safety Instructions and Passing Them on to Others.....	16
3.2.2 Requirements for Safe Use.....	16
3.2.3 Hazards by Improper Use.....	17
3.3 Instructions with Regard to Specific Dangers.....	19
3.3.1 Protection Against Contact With Electrical Parts and Housings.....	19
3.3.2 Protective Extra-Low Voltage as Protection Against Electric Shock	20
3.3.3 Protection Against Dangerous Movements.....	21
3.3.4 Protection Against Magnetic and Electromagnetic Fields During Operation and Mounting.....	22
3.3.5 Protection Against Contact With Hot Parts.....	22
3.3.6 Protection During Handling and Mounting.....	23
3.3.7 Battery Safety.....	23
3.3.8 Protection Against Pressurized Systems.....	24
3.4 Explanation of Signal Words and the Safety Alert Symbol.....	24
4 Overview.....	27
4.1 Overview.....	27
4.2 Visualization devices.....	27
4.2.1 Overview.....	27
4.2.2 VCP.....	27
Brief Description.....	27
Technical Data.....	29
Models.....	29
Accessories.....	30
Documentation.....	31
4.2.3 VCH.....	31
Brief Description.....	31

Table of Contents

	Page
Field of Application.....	32
Technical Data.....	32
Models.....	32
Accessories.....	33
Documentation.....	33
4.2.4 VEP.....	33
Brief Description.....	33
Brief Description.....	33
Technical Data.....	34
Wear Parts.....	34
Models.....	34
Accessories.....	36
Documentation.....	36
4.2.5 VEH.....	36
Brief Description.....	36
Field of Application.....	37
Technical Data.....	37
Wear Parts.....	38
Models.....	38
Brief Description.....	38
4.2.6 VSx and VDP.....	38
Brief Description.....	38
Field of Application.....	39
Technical Data.....	39
Wear Parts.....	39
Models.....	40
Accessories.....	40
Documentation.....	41
4.2.7 VPx and VDP.....	41
Brief Description.....	41
Field of Application.....	42
Technical Data.....	42
Wear Parts.....	42
Models.....	43
Accessories.....	44
Documentation.....	45
4.2.8 VPP.....	45
Brief Description.....	45
Field of Application.....	46
Technical Data.....	46
Wear Parts.....	47
Models.....	47
Accessories.....	47
Documentation.....	48
4.2.9 VSP.....	48
Brief Description.....	48

Table of Contents

	Page
Field of Application.....	49
Technical Data.....	49
Wear Parts.....	49
Models.....	50
Accessories.....	50
Documentation.....	50
4.3 Visualizations.....	50
4.3.1 Overview.....	50
4.3.2 VI-Logic.....	51
Field of Application.....	51
Properties of the Integrated Visualization.....	51
Visualization Objects.....	52
Supported Visualization Devices.....	53
Protocol.....	53
Documentation.....	53
4.3.3 VI Composer.....	53
Field of Application.....	53
Configurable Small Operating Terminals.....	53
Communication Protocol BRC-Symbolic.....	54
Connection Settings.....	54
Documentation.....	55
4.3.4 WinStudio.....	55
Overview.....	55
WinStudio Elements in the Project Explorer.....	56
General WinStudio Settings and Functions.....	57
WinStudio Screens.....	65
WinStudio Variables.....	67
Linked Symbols.....	68
WinStudio Tasks.....	70
WinStudio Tools.....	77
Database Spy.....	78
LogWin.....	79
Cross Reference Search.....	79
Variable Import Wizard.....	80
Export/Import.....	80
WinStudio Licensing.....	81
Multilingualism.....	83
WinStudio Applications.....	83
Further Information.....	84
4.4 Communication Interfaces.....	84
4.4.1 Overview.....	84
4.4.2 OPC.lwSCP.....	85
Properties.....	85
SCP-OPC Configurator.....	85
lwSCP and OPC.lwSCP Items.....	89
Disclaimer.....	92

Table of Contents

	Page
4.4.3 BRWS Driver (Bosch Rexroth WinStudio Driver).....	92
Properties.....	92
Application Area.....	92
Configuration.....	92
Driver Sheets.....	94
Protocol Structure.....	96
Volume of Data.....	96
PLC Variable and Parameter Support.....	96
Error Messages.....	99
Message Information.....	99
Documentation.....	100
4.4.4 IndraLogic OPC.DA.....	100
Properties.....	100
Configuring the IndraLogic OPC.DA Server.....	101
Disclaimer.....	111
4.4.5 OPC UA.....	111
Overview.....	111
Introducing the OPC Unified Architecture.....	112
Bosch Rexroth OPC UA Server.....	117
OPC UA Communication Sample.....	119
4.4.6 Modbus TCP.....	127
Introduction and Overview.....	127
5 Application Samples.....	129
5.1 Overview.....	129
5.2 Symbol Configuration.....	129
5.2.1 General.....	129
5.2.2 Symbol Configuration Using a Wizard.....	130
5.2.3 Symbol Configuration Using a Pragma.....	131
5.3 VI Composer.....	132
5.4 WinStudio with IndraLogic XLC.....	133
5.4.1 General.....	133
5.4.2 Technical Requirements.....	133
5.4.3 Creating the IndraWorks Project.....	133
5.4.4 Integrating the WinStudio Visualization in IndraWorks.....	140
5.4.5 Creating the WinStudio Visualization.....	148
5.4.6 Starting the Interface.....	156
5.5 IndraLogic Visualization.....	160
5.6 HMIs of 3rd Party Manufacturers.....	160
5.6.1 OPC.DA Server.....	160
Tools to Configure the OPC.DA Server on a PC.....	160
Tools to Transfer the OPC.DA Server Configuration to a Visualization Device VEPxx.4 with WindowsCE.....	161
5.6.2 ModbusTCP.....	162
General.....	162
Configuration with Siemens WinCC-flexible 2007.....	163

Table of Contents

	Page
Configuration with Pro-face GP-Pro EX 2.2.....	167
6 Client Development.....	175
6.1 OPC Client Visual Basic .NET.....	175
6.1.1 General.....	175
6.1.2 Visual Basic Project "OPC Client".....	175
New Project.....	175
Adding References.....	176
Adding Windows Form and Control Element.....	177
Implementing OPC Client.....	179
6.2 OPC Client Visual C# .NET.....	197
6.2.1 General.....	197
6.2.2 Visual C# Project "OPC Client ".....	198
New Project.....	198
Adding References.....	198
Adding Windows Form and Control Elements.....	199
Implementing OPC Client.....	201
7 Service and Support.....	219
Index.....	221

1 About this Documentation

1.1 Validity of the Documentation

- Target group** This documentation is directed at users wanting to integrate the supported visualization systems of IndraLogic XLC and IndraMotion MLC/MLP.
- This documentation supports the user during all phases
- Selection
 - Engineering and
 - Commissioning

1.2 Required and Supplementing Documentations

Documentation titles with type designation codes and parts numbers

IndraWorks		MLC	XLC
/36/	Rexroth IndraWorks 12VRS Software Installation DOK-IWORKS-SOFTINS*V12-COxx-EN-P, R911334396 This documentation describes the IndraWorks installation.	X	X
/5/	Rexroth IndraWorks 12VRS Engineering DOK-IWORKS-ENGINEE*V12-APxx-EN-P, R911334388 This documentation describes the application of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.	X	X
/20/	Rexroth IndraMotion MLC 12VRS Functional Description DOK-MLC***-FUNC****V12-APxx-EN-P, R911333848 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.	X	
/20/	Rexroth IndraLogic XLC 12VRS Functional Description DOK-XLC***-FUNC****V12-APxx-EN-P, R911333878 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraLogic XLC.		X
/7/	Rexroth IndraWorks 12VRS CamBuilder DOK-IWORKS-CAMBUIL*V12-APxx-EN-P, R911333842 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.	X	X
/37/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Automation Interface DOK-XLCMLC-AUT*INT*V12-APxx-EN-P, R911334178 This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.	X	X
/38/	Rexroth IndraWorks 12VRS FDT Container DOK-IWORKS-FDT*CON*V12-APxx-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.	X	X

About this Documentation

/29/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Project Conversion DOK-XLCMLC-PROCONV*V12-APxx-EN-P, R911334187 This documentation described the project conversion of IndraLogic 04VRS and IndraMotion MLC04VRS on IndraWorks Version 12 with IndraLogic 2G. It especially focuses on changes in the field of Motion and PLC.	X	X
/28/	Rexroth IndraMotion MLC 12VRS Commissioning DOK-MLC***-STARTUP*V12-COxx-EN-P, R911333858 This documentation describes the steps required for commissioning and performing service on the IndraMotion MLC system. It includes checklists for tasks to be frequently performed and a detailed description of the steps.	X	

Motion		MLC	XLC
/23/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS PLCopen Libraries DOK-XLCMLC-FUNLIB**V12-LIxx-EN-P, R911334182 This documentation describes the function blocks, functions and data types of the RIL_CommonTypes, ML_Base and ML_PLCOpen libraries for the IndraLogic XLC/IndraMotion MLC. It also includes the error reactions of function blocks.	X	X
/27/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Generic Application Template DOK-XLCMLC-TF*GAT**V12-APxx-EN-P, R911334191 This documentation provides a structured template to the IndraLogic PLC programmer. This template can be used to add and edit the PLC programming code. It includes the template, the template wizard and example applications.	X	X
/31/	Rexroth IndraMotion MLC 12VRS RCL Programming Instruction DOK-MLC***-RCL*PRO*V12-APxx-EN-P, R911333852 This documentation provides information on the RobotControl. It is given most importance to the programming language RCL (RobotControl Language). The program structure, variables, functions, motion statements and the required system parameters are described.	X	
/21/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Parameters DOK-XLCMLC-PARAM***V12-RExx-EN-P, R911334176 This documentation describes the parameters of the XLC/MLC systems as well as the interaction between parameterization and programming. It includes the axis parameters, control parameters, kinematic parameters, touch probe parameters and programmable limit switch parameters.	X	X
/10/	Rexroth IndraDrive; Firmware for Drive Controllers MPH, MPB, MPD, MPC-07 DOK-INDRV*-MP*-07VRS**-FKxx-EN-P, R911328670		
/11/	Rexroth IndraDrive MPx-16 Functions DOK-INDRV*-MP*-16VRS**-APxx-EN-P, R911326767		

Field buses		MLC	XLC
/39/	Rexroth IndraMotion MLC 11VRS PLCopen Field Bus DOK-IM*ML*-PLCFBUS*V11-APxx-EN-P, R911333896 This documentation describes the creation of field bus drives in an IndraWorks project, function blocks, functions and data types of the libraries RIL_CommonTypes.library (excerpt for field bus drives), RMB_PLCOpenFieldBus.library, RIL_Utility.library (excerpt for field bus drives). It also includes the error reactions of function blocks.	X	X
/4/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Field Buses DOK-XLCMLC-FB*****V12-APxx-EN-P, R911334394 This documentation describes the supported field buses and their diagnostic function blocks.	X	X

About this Documentation

HMI		MLC	XLC
/8/	Rexroth IndraWorks 12VRS HMI DOK-IWORKS-HMI*****V12-APxx-EN-P, R911334392 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.	X	X
/6/	Rexroth IndraWorks 12VRS WinStudio DOK-IWORKS-WINSTUD*V12-APxx-EN-P, R911333844 This documentation describes the installation of the software, working with WinStudio and the creation and operation of applications.	X	X
/50/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS HMI Connection DOK-XLCMLC-HMI*****V12-APxx-EN-P, R911334184 This documentation describes the visualization systems supported by the IndraLogic XLC and IndraMotion MLC and their connection.	X	X

PLC		MLC	XLC
/3/	Rexroth IndraWorks 12VRS IndraLogic 2G Programming Instruction DOK-IWORKS-IL2GPRO*V12-APxx-EN-P, R911334390 This documentation describes the PLC programming tool IndraLogic 2G and its usage. It includes the basic usage, first steps, visualization, menu items and editors.	X	X
/33/	Rexroth IndraWorks 12VRS, Basic Libraries, IndraLogic 2G DOK-IL*2G*-BASLIB**V12-Llxx-EN-P, R911333835 This documentation describes the system-comprehensive PLC libraries.	X	X

Technology			
/30/	Rexroth IndraMotion MLC 12VRS Technology Libraries DOK-MLC***-TF*LIB**V12-Llxx-EN-P, R911333868 This documentation describes the function blocks, functions and data types of the libraries "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" and "ML_TechBase.library". It also includes libraries for the winder functionality, register controller functionality and CrossCutter functionality.	X	
/60/	Rexroth IndraMotion MLC 12VRS RegisterControl (Library) DOK-MLC***-REGI*CO*V12-Llxx-EN-P, R911333856 This documentation describes the inputs and outputs of the individual function blocks and provides notes on their usage.	X	
/62/	Rexroth IndraMotion MLC 12VRS RegisterControl (Application Manual) DOK-MLC***-REGI*CO*V12-APxx-EN-P, R911333854 This documentation describes the application of the integrated register control for a rotogravure printing machine. The components of the mark stream sensor, the HMI application and the error recovery options are described. This instruction provides information on how to operate the register control, react on errors and query diagnostics. This documentation is written for machine setters and machine operators.	X	
/49/	Rexroth IndraMotion MLC 12VRS Application of Winder Functions DOK-MLC***-TF*WIND*V12-APxx-EN-P, R911333870 This application-related system documentation describes the usage of the winder technology functions.	X	

About this Documentation

Hardware		MLC	XLC
/1/	Rexroth IndraControl L45/L65 DOK-CONTRL-IC*L45*L65*-PRxx-EN-P, R911324661	X	X
/2/	Rexroth IndraControl L25 DOK-CONTRL-IC*L25*****-PRxx-EN-P, R911328474	X	X
/24/	Rexroth IndraControl Lxx 12VRS Function Modules DOK-CONTRL-FM*LXX**V12-APxx-EN-P, R911333830 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.	X	X
/12/	Rexroth IndraDrive Drive Controllers MPx-02 to MPx-07 DOK-INDRV*-GEN-**VRS**-PAxx-EN-P, R911297317		
/13/	Rexroth IndraDrive MPx-02 to MPx-07 and HMV DOK-INDRV*-GEN-**VRS**-WAxx-EN-P, R911297319		
/35/	Rexroth IndraDrive Drive Controller Control Sections CSB01, CSH01, CDB01 DOK-INDRV*-CSH*****-PR08-EN-P, R911295012		

Diagnostics and Service		MLC	XLC
/26/	Rexroth IndraMotion MLC/XLC 11VRS Service Tool DOK-IM*ML*-IMST***V11-RExx-EN-P, R911331940	X	X
/22/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Diagnostics DOK-XLCMLC-DIAG***V12-RExx-EN-P, R911334180 This documentation includes all control parameters implemented in the control systems IndraLogic XLC and IndraMotion MLC.	X	X

System Overview		MLC	XLC
/48/	Rexroth IndraMotion for Printing 12VRS System Overview DOK-IM*PR*-SYSTEM**V11-PRxx-EN-P, R911333840 This documentation describes the product IndraMotion for Packaging. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	
/48/	Rexroth IndraMotion for Packaging 12VRS System Overview DOK-IM*PA*-SYSTEM**V12-PRxx-EN-P, R911333838 This documentation describes the product IndraMotion for Packaging. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	
/9/	Rexroth IndraMotion MLC 12VRS System Overview DOK-MLC***-SYSTEM**V12-PRxx-EN-P, R911333860 This documentation provides an overview on the possible hardware/software components of the automation system IndraMotion MLC of the named version. It helps assembling a system.	X	
/9/	Rexroth IndraLogic XLC 12VRS System Overview DOK-XLC***-SYSTEM**V12-PRxx-EN-P, R911333880 This documentation provides an overview on the possible hardware/software components of the automation system IndraLogic XLC of the named version. It helps assembling a system.		X

About this Documentation

First Steps		MLC	XLC
/25/	Rexroth IndraMotion MLC 12VRS First Steps DOK-MLC***-F*STEP**V12-COxx-EN-P, R911333846 This documentation describes the first steps of the IndraMotion MLC and the RobotControl. It includes the hardware and software prerequisites as well as the creation of a project.	X	
/25/	Rexroth IndraLogic XLC 12VRS First Steps DOK-XLC***-F*STEP**V12-COxx-EN-P, R911333876 This documentation describes the first steps of the IndraLogic XLC. It includes the hardware and software prerequisites as well as the creation of a project.		X

xx Respective edition
Fig. 1-1: XCL/MLC documentation overview

1.3 Information Representation

1.3.1 Safety Instructions

The safety instructions, if contained in the documentation, contain certain signal words (Danger, Warning, Caution, Note) and sometimes a signal symbol (according to ANSI Z535.6-2006).

The signal word draws the attention to the safety note and indicates the risk potential.

The signal symbol (triangular safety reflector with three exclamation marks), preceding the signal words Danger, Warning, Caution indicates a hazard for persons.

DANGER

In case of non-compliance with these safety instructions, death or serious injury **will** occur.

WARNING

In case of non-compliance with these safety instructions, death or serious injury **can** occur.

CAUTION

In case of non-compliance with these safety instructions, minor or moderately severe injury can occur.

NOTICE

In case of non-compliance with these safety instructions, material or property damage can occur.

1.3.2 Symbols Used

Note Notes are represented as follows:



This is a note for the user.

Tip Tips are represented as follows:

About this Documentation



This is a tip for the user.

1.3.3 Names and Abbreviations

Term	Explanation
CANOpen	Field bus
DeviceNet	Field bus
Ethernet	Communication interface
IWE	IndraWorks Engineering
IWO	IndraWorks Operation
LWL	Optical guide
NC	Numerical Control
OEM	Original Equipment Manufacturer
PROFIBUS-DP	Field bus
sercos	Field bus

Fig. 1-2: Used terms and abbreviations

2 Important Instructions for Use

2.1 Intended Use

2.1.1 Introduction

Bosch Rexroth products are developed and manufactured according to the . Before being delivered, they are tested regarding their operating safety.

WARNING

Personal injury and damage to property due to wrong use of the products!

The products were developed for the use in an industrial environment and must only be used as intended. If they are not used as intended, situations may arise that can lead to personal injury or damage to property.



Bosch Rexroth, as the manufacturer of the products, will not assume any warranty, liability or payment of damages in case of damage resulting from a non-intended use of the products. If he fails to use the products as intended, the user will be solely responsible for any resulting risks.

Before you use Bosch Rexroth products, the following requirements must be met in order to guarantee the intended use of the products:

- Any person who in any way whatsoever is involved in handling our products must read and understand the corresponding safety instructions and the notes on intended use.
- To the extent that the products are hardware elements, they must be left in their original state; i.e. no structural modifications must be carried out. Software products must not be decompiled and the source codes must not be modified.
- Damaged or defective products must neither be installed nor be put into operation.
- It must be guaranteed that the products are installed according to the instructions mentioned in the documentation.

2.1.2 Scope of Application

The IndraMotion MLC, IndraMotion MLP and IndraMotion MLD by Bosch Rexroth is intended for Motion/Logic applications.



The IndraMotion MLC, MLP and MLD only be used together with the accessories and attachment parts given in this documentation. Components that are not expressly mentioned must neither be attached nor connected. The same is valid for cables and lines.

The operation must only be carried out with the component configurations and combinations that were expressly mentioned and with the software and firmware indicated and specified in the respective functional description.

The IndraMotion MLC, MLP and MLD have been developed for use in single and multi-axis drive and control tasks.

Important Instructions for Use

To allow for application-specific requirements in IndraMotion MLC, MLP and MLD, our product range comprises various device types with different power and interfaces.

Typical applications for IndraMotion MLC, MLP and MLD are:

- Handling and mounting systems
- Packaging and food machines
- Printing and paper-processing machines
- Machine tools

The IndraMotion MLC, MLP and MLD must be operated only in the mounting and installation conditions, the position, and the ambient conditions (temperature, degree of protection, moisture, EMC, etc.) specified in this Documentation.

2.2 Improper Use

Use of the IndraMotion MLC, MLP and MLD in applications other than those specified or described in the documentation and technical data is considered as "improper".

The IndraMotion MLC, MLP and MLD must not be used if

- they are subjected to operating conditions not corresponding to the specified ambient conditions. Operation under water, under extreme temperature fluctuations or extreme maximal temperatures etc. is prohibited.
- the intended applications were not expressly validated by Bosch Rexroth. Therefore, read the statements given in the general safety instructions!

3 Safety Instructions for Electric Drives and Controls

3.1 Definitions of Terms

Application Documentation	Application documentation comprises the entire documentation used to inform the user of the product about the use and safety-relevant features for configuring, integrating, installing, mounting, commissioning, operating, maintaining, repairing and decommissioning the product. The following terms are also used for this kind of documentation: User Guide, Operation Manual, Commissioning Manual, Instruction Manual, Project Planning Manual, Application Manual, etc.
Component	A component is a combination of elements with a specified function, which are part of a piece of equipment, device or system. Components of the electric drive and control system are, for example, supply units, drive controllers, mains choke, mains filter, motors, cables, etc.
Control System	A control system comprises several interconnected control components placed on the market as a single functional unit.
Device	A device is a finished product with a defined function, intended for users and placed on the market as an individual piece of merchandise.
Electrical Equipment	Electrical equipment encompasses all devices used to generate, convert, transmit, distribute or apply electrical energy, such as electric motors, transformers, switching devices, cables, lines, power-consuming devices, circuit board assemblies, plug-in units, control cabinets, etc.
Electric Drive System	An electric drive system comprises all components from mains supply to motor shaft; this includes, for example, electric motor(s), motor encoder(s), supply units and drive controllers, as well as auxiliary and additional components, such as mains filter, mains choke and the corresponding lines and cables.
Installation	An installation consists of several devices or systems interconnected for a defined purpose and on a defined site which, however, are not intended to be placed on the market as a single functional unit.
Machine	A machine is the entirety of interconnected parts or units at least one of which is movable. Thus, a machine consists of the appropriate machine drive elements, as well as control and power circuits, which have been assembled for a specific application. A machine is, for example, intended for processing, treatment, movement or packaging of a material. The term "machine" also covers a combination of machines which are arranged and controlled in such a way that they function as a unified whole.
Manufacturer	The manufacturer is an individual or legal entity bearing responsibility for the design and manufacture of a product which is placed on the market in the individual's or legal entity's name. The manufacturer can use finished products, finished parts or finished elements, or contract out work to subcontractors. However, the manufacturer must always have overall control and possess the required authority to take responsibility for the product.
Product	Examples of a product: Device, component, part, system, software, firmware, among other things.
Project Planning Manual	A project planning manual is part of the application documentation used to support the sizing and planning of systems, machines or installations.
Qualified Persons	In terms of this application documentation, qualified persons are those persons who are familiar with the installation, mounting, commissioning and operation of the components of the electric drive and control system, as well as with the hazards this implies, and who possess the qualifications their work

Safety Instructions for Electric Drives and Controls

requires. To comply with these qualifications, it is necessary, among other things,

- 1) to be trained, instructed or authorized to switch electric circuits and devices safely on and off, to ground them and to mark them
- 2) to be trained or instructed to maintain and use adequate safety equipment
- 3) to attend a course of instruction in first aid

User A user is a person installing, commissioning or using a product which has been placed on the market.

3.2 General Information

3.2.1 Using the Safety Instructions and Passing Them on to Others

Do not attempt to install and operate the components of the electric drive and control system without first reading all documentation provided with the product. Read and understand these safety instructions and all user documentation prior to working with these components. If you do not have the user documentation for the components, contact your responsible Bosch Rexroth sales partner. Ask for these documents to be sent immediately to the person or persons responsible for the safe operation of the components.

If the component is resold, rented and/or passed on to others in any other form, these safety instructions must be delivered with the component in the official language of the user's country.

Improper use of these components, failure to follow the safety instructions in this document or tampering with the product, including disabling of safety devices, could result in property damage, injury, electric shock or even death.

3.2.2 Requirements for Safe Use

Read the following instructions before initial commissioning of the components of the electric drive and control system in order to eliminate the risk of injury and/or property damage. You must follow these safety instructions.

- Bosch Rexroth is not liable for damages resulting from failure to observe the safety instructions.
- Read the operating, maintenance and safety instructions in your language before commissioning. If you find that you cannot completely understand the application documentation in the available language, please ask your supplier to clarify.
- Proper and correct transport, storage, mounting and installation, as well as care in operation and maintenance, are prerequisites for optimal and safe operation of the component.
- Only qualified persons may work with components of the electric drive and control system or within its proximity.
- Only use accessories and spare parts approved by Bosch Rexroth.
- Follow the safety regulations and requirements of the country in which the components of the electric drive and control system are operated.
- Only use the components of the electric drive and control system in the manner that is defined as appropriate. See chapter "Appropriate Use".
- The ambient and operating conditions given in the available application documentation must be observed.
- Applications for functional safety are only allowed if clearly and explicitly specified in the application documentation "Integrated Safety Technolo-

Safety Instructions for Electric Drives and Controls

gy". If this is not the case, they are excluded. Functional safety is a safety concept in which measures of risk reduction for personal safety depend on electrical, electronic or programmable control systems.

- The information given in the application documentation with regard to the use of the delivered components contains only examples of applications and suggestions.

The machine and installation manufacturers must

- make sure that the delivered components are suited for their individual application and check the information given in this application documentation with regard to the use of the components,
- make sure that their individual application complies with the applicable safety regulations and standards and carry out the required measures, modifications and complements.
- Commissioning of the delivered components is only allowed once it is sure that the machine or installation in which the components are installed complies with the national regulations, safety specifications and standards of the application.
- Operation is only allowed if the national EMC regulations for the application are met.
- The instructions for installation in accordance with EMC requirements can be found in the section on EMC in the respective application documentation.

The machine or installation manufacturer is responsible for compliance with the limit values as prescribed in the national regulations.

- The technical data, connection and installation conditions of the components are specified in the respective application documentations and must be followed at all times.

National regulations which the user must take into account

- European countries: In accordance with European EN standards
- United States of America (USA):
 - National Electrical Code (NEC)
 - National Electrical Manufacturers Association (NEMA), as well as local engineering regulations
 - Regulations of the National Fire Protection Association (NFPA)
- Canada: Canadian Standards Association (CSA)
- Other countries:
 - International Organization for Standardization (ISO)
 - International Electrotechnical Commission (IEC)

3.2.3 Hazards by Improper Use

- High electrical voltage and high working current! Danger to life or serious injury by electric shock!
- High electrical voltage by incorrect connection! Danger to life or injury by electric shock!
- Dangerous movements! Danger to life, serious injury or property damage by unintended motor movements!
- Health hazard for persons with heart pacemakers, metal implants and hearing aids in proximity to electric drive systems!

Safety Instructions for Electric Drives and Controls

- Risk of burns by hot housing surfaces!
- Risk of injury by improper handling! Injury by crushing, shearing, cutting, hitting!
- Risk of injury by improper handling of batteries!
- Risk of injury by improper handling of pressurized lines!

3.3 Instructions with Regard to Specific Dangers

3.3.1 Protection Against Contact With Electrical Parts and Housings



This section concerns components of the electric drive and control system with voltages of **more than 50 volts**.

Contact with parts conducting voltages above 50 volts can cause personal danger and electric shock. When operating components of the electric drive and control system, it is unavoidable that some parts of these components conduct dangerous voltage.

High electrical voltage! Danger to life, risk of injury by electric shock or serious injury!

- Only qualified persons are allowed to operate, maintain and/or repair the components of the electric drive and control system.
- Follow the general installation and safety regulations when working on power installations.
- Before switching on, the equipment grounding conductor must have been permanently connected to all electric components in accordance with the connection diagram.
- Even for brief measurements or tests, operation is only allowed if the equipment grounding conductor has been permanently connected to the points of the components provided for this purpose.
- Before accessing electrical parts with voltage potentials higher than 50 V, you must disconnect electric components from the mains or from the power supply unit. Secure the electric component from reconnection.
- With electric components, observe the following aspects:
Always wait **30 minutes** after switching off power to allow live capacitors to discharge before accessing an electric component. Measure the electrical voltage of live parts before beginning to work to make sure that the equipment is safe to touch.
- Install the covers and guards provided for this purpose before switching on.
- Never touch electrical connection points of the components while power is turned on.
- Do not remove or plug in connectors when the component has been powered.
- Under specific conditions, electric drive systems can be operated at mains protected by residual-current-operated circuit-breakers sensitive to universal current (RCDs/RCMs).
- Secure built-in devices from penetrating foreign objects and water, as well as from direct contact, by providing an external housing, for example a control cabinet.

High housing voltage and high leakage current! Danger to life, risk of injury by electric shock!

- Before switching on and before commissioning, ground or connect the components of the electric drive and control system to the equipment grounding conductor at the grounding points.

Safety Instructions for Electric Drives and Controls

- Connect the equipment grounding conductor of the components of the electric drive and control system permanently to the main power supply at all times. The leakage current is greater than 3.5 mA.
- Establish an equipment grounding connection with a minimum cross section according to the table below. With an outer conductor cross section smaller than 10 mm² (8 AWG), the alternative connection of two equipment grounding conductors is allowed, each having the same cross section as the outer conductors.

Cross section outer conductor	Minimum cross section equipment grounding conductor Leakage current ≥ 3.5 mA	
	1 equipment grounding conductor	2 equipment grounding conductors
1,5 mm ² (AWG 16)	10 mm ² (AWG 8)	2 × 1,5 mm ² (AWG 16)
2,5 mm ² (AWG 14)		2 × 2,5 mm ² (AWG 14)
4 mm ² (AWG 12)		2 × 4 mm ² (AWG 12)
6 mm ² (AWG 10)		2 × 6 mm ² (AWG 10)
10 mm ² (AWG 8)		-
16 mm ² (AWG 6)	16 mm ² (AWG 6)	-
25 mm ² (AWG 4)		-
35 mm ² (AWG 2)		-
50 mm ² (AWG 1/0)	25 mm ² (AWG 4)	-
70 mm ² (AWG 2/0)	35 mm ² (AWG 2)	-
...	...	...

Fig.3-1: Minimum Cross Section of the Equipment Grounding Connection

3.3.2 Protective Extra-Low Voltage as Protection Against Electric Shock

Protective extra-low voltage is used to allow connecting devices with basic insulation to extra-low voltage circuits.

On components of an electric drive and control system provided by Bosch Rexroth, all connections and terminals with voltages between 5 and 50 volts are PELV ("Protective Extra-Low Voltage") systems. It is allowed to connect devices equipped with basic insulation (such as programming devices, PCs, notebooks, display units) to these connections.

Danger to life, risk of injury by electric shock! High electrical voltage by incorrect connection!

If extra-low voltage circuits of devices containing voltages and circuits of more than 50 volts (e.g., the mains connection) are connected to Bosch Rexroth products, the connected extra-low voltage circuits must comply with the requirements for PELV ("Protective Extra-Low Voltage").

3.3.3 Protection Against Dangerous Movements

Dangerous movements can be caused by faulty control of connected motors. Some common examples are:

- Improper or wrong wiring or cable connection
- Operator errors
- Wrong input of parameters before commissioning
- Malfunction of sensors and encoders
- Defective components
- Software or firmware errors

These errors can occur immediately after equipment is switched on or even after an unspecified time of trouble-free operation.

The monitoring functions in the components of the electric drive and control system will normally be sufficient to avoid malfunction in the connected drives. Regarding personal safety, especially the danger of injury and/or property damage, this alone cannot be relied upon to ensure complete safety. Until the integrated monitoring functions become effective, it must be assumed in any case that faulty drive movements will occur. The extent of faulty drive movements depends upon the type of control and the state of operation.

Dangerous movements! Danger to life, risk of injury, serious injury or property damage!

A **risk assessment** must be prepared for the installation or machine, with its specific conditions, in which the components of the electric drive and control system are installed.

As a result of the risk assessment, the user must provide for monitoring functions and higher-level measures on the installation side for personal safety. The safety regulations applicable to the installation or machine must be taken into consideration. Unintended machine movements or other malfunctions are possible if safety devices are disabled, bypassed or not activated.

To avoid accidents, injury and/or property damage:

- Keep free and clear of the machine's range of motion and moving machine parts. Prevent personnel from accidentally entering the machine's range of motion by using, for example:
 - Safety fences
 - Safety guards
 - Protective coverings
 - Light barriers
- Make sure the safety fences and protective coverings are strong enough to resist maximum possible kinetic energy.
- Mount emergency stopping switches in the immediate reach of the operator. Before commissioning, verify that the emergency stopping equipment works. Do not operate the machine if the emergency stopping switch is not working.
- Prevent unintended start-up. Isolate the drive power connection by means of OFF switches/OFF buttons or use a safe starting lockout.
- Make sure that the drives are brought to safe standstill before accessing or entering the danger zone.

Safety Instructions for Electric Drives and Controls

- Additionally secure vertical axes against falling or dropping after switching off the motor power by, for example,
 - mechanically securing the vertical axes,
 - adding an external braking/arrester/clamping mechanism or
 - ensuring sufficient counterbalancing of the vertical axes.
- The standard equipment **motor holding brake** or an external holding brake controlled by the drive controller is **not sufficient to guarantee personal safety!**
- Disconnect electrical power to the components of the electric drive and control system using the master switch and secure them from reconnection ("lock out") for:
 - Maintenance and repair work
 - Cleaning of equipment
 - Long periods of discontinued equipment use
- Prevent the operation of high-frequency, remote control and radio equipment near components of the electric drive and control system and their supply leads. If the use of these devices cannot be avoided, check the machine or installation, at initial commissioning of the electric drive and control system, for possible malfunctions when operating such high-frequency, remote control and radio equipment in its possible positions of normal use. It might possibly be necessary to perform a special electromagnetic compatibility (EMC) test.

3.3.4 Protection Against Magnetic and Electromagnetic Fields During Operation and Mounting

Magnetic and electromagnetic fields generated by current-carrying conductors or permanent magnets of electric motors represent a serious danger to persons with heart pacemakers, metal implants and hearing aids.

Health hazard for persons with heart pacemakers, metal implants and hearing aids in proximity to electric components!

- Persons with heart pacemakers and metal implants are not allowed to enter the following areas:
 - Areas in which components of the electric drive and control systems are mounted, commissioned and operated.
 - Areas in which parts of motors with permanent magnets are stored, repaired or mounted.
- If it is necessary for somebody with a heart pacemaker to enter such an area, a doctor must be consulted prior to doing so. The noise immunity of implanted heart pacemakers differs so greatly that no general rules can be given.
- Those with metal implants or metal pieces, as well as with hearing aids, must consult a doctor before they enter the areas described above.

3.3.5 Protection Against Contact With Hot Parts

Hot surfaces of components of the electric drive and control system. Risk of burns!

Safety Instructions for Electric Drives and Controls

- Do not touch hot surfaces of, for example, braking resistors, heat sinks, supply units and drive controllers, motors, windings and laminated cores!
- According to the operating conditions, temperatures of the surfaces can be **higher than 60 °C** (140 °F) during or after operation.
- Before touching motors after having switched them off, let them cool down for a sufficient period of time. Cooling down can require **up to 140 minutes!** The time required for cooling down is approximately five times the thermal time constant specified in the technical data.
- After switching chokes, supply units and drive controllers off, wait **15 minutes** to allow them to cool down before touching them.
- Wear safety gloves or do not work at hot surfaces.
- For certain applications, and in accordance with the respective safety regulations, the manufacturer of the machine or installation must take measures to avoid injuries caused by burns in the final application. These measures can be, for example: Warnings at the machine or installation, guards (shieldings or barriers) or safety instructions in the application documentation.

3.3.6 Protection During Handling and Mounting

Risk of injury by improper handling! Injury by crushing, shearing, cutting, hitting!

- Observe the relevant statutory regulations of accident prevention.
- Use suitable equipment for mounting and transport.
- Avoid jamming and crushing by appropriate measures.
- Always use suitable tools. Use special tools if specified.
- Use lifting equipment and tools in the correct manner.
- Use suitable protective equipment (hard hat, safety goggles, safety shoes, safety gloves, for example).
- Do not stand under hanging loads.
- Immediately clean up any spilled liquids from the floor due to the risk of falling!

3.3.7 Battery Safety

Batteries consist of active chemicals in a solid housing. Therefore, improper handling can cause injury or property damage.

Risk of injury by improper handling!

- Do not attempt to reactivate low batteries by heating or other methods (risk of explosion and cauterization).
- Do not attempt to recharge the batteries as this may cause leakage or explosion.
- Do not throw batteries into open flames.
- Do not dismantle batteries.
- When replacing the battery/batteries, do not damage the electrical parts installed in the devices.
- Only use the battery types specified for the product.

Safety Instructions for Electric Drives and Controls



Environmental protection and disposal! The batteries contained in the product are considered dangerous goods during land, air, and sea transport (risk of explosion) in the sense of the legal regulations. Dispose of used batteries separately from other waste. Observe the national regulations of your country.

3.3.8 Protection Against Pressurized Systems

According to the information given in the Project Planning Manuals, motors and components cooled with liquids and compressed air can be partially supplied with externally fed, pressurized media, such as compressed air, hydraulics oil, cooling liquids and cooling lubricants. Improper handling of the connected supply systems, supply lines or connections can cause injuries or property damage.

Risk of injury by improper handling of pressurized lines!

- Do not attempt to disconnect, open or cut pressurized lines (risk of explosion).
- Observe the respective manufacturer's operating instructions.
- Before dismounting lines, relieve pressure and empty medium.
- Use suitable protective equipment (safety goggles, safety shoes, safety gloves, for example).
- Immediately clean up any spilled liquids from the floor due to the risk of falling!



Environmental protection and disposal! The agents (e.g., fluids) used to operate the product might not be environmentally friendly. Dispose of agents harmful to the environment separately from other waste. Observe the national regulations of your country.

3.4 Explanation of Signal Words and the Safety Alert Symbol

The Safety Instructions in the available application documentation contain specific signal words (DANGER, WARNING, CAUTION or NOTICE) and, where required, a safety alert symbol (in accordance with ANSI Z535.6-2006).

The signal word is meant to draw the reader's attention to the safety instruction and identifies the hazard severity.

The safety alert symbol (a triangle with an exclamation point), which precedes the signal words DANGER, WARNING and CAUTION, is used to alert the reader to personal injury hazards.



In case of non-compliance with this safety instruction, death or serious injury will occur.

Safety Instructions for Electric Drives and Controls

⚠ WARNING

In case of non-compliance with this safety instruction, death or serious injury **could** occur.

⚠ CAUTION

In case of non-compliance with this safety instruction, minor or moderate injury could occur.

NOTICE

In case of non-compliance with this safety instruction, property damage could occur.

4 Overview

4.1 Overview

4.2 Visualization devices

4.2.1 Overview

Rexroth IndraControl V is the comprehensive HMI device portfolio for individual control, operation and observation in all industrial sectors. With scalable hardware and software, IndraControl V can be precisely adapted to your machine-specific requirements.

IndraControl V integrates all necessary functions for efficient automation - starting with convenient operation and clear and transparent visualization right through to integrated controls and diagnostics. Used in conjunction with Rexroth system solutions that have been tested in practice, you therefore obtain a complete automation solution for:

- Transfer machinery
- Printing and paper-processing machinery
- Food and packaging machinery
- Metal forming machinery
- Wood working machinery
- Textile machinery
- Handling and mounting systems

The following models of IndraControl V are available:

- Controller-based
 - IndraControl VCP ([chapter 4.2.2 "VCP" on page 27](#))
 - IndraControl VCH ([chapter 4.2.3 "VCH" on page 31](#))
- Embedded PC-based
 - IndraControl VEP ([chapter 4.2.4 "VEP" on page 33](#))
 - IndraControl VEH ([chapter 4.2.5 "VEH" on page 36](#))
- PC-based
 - IndraControl VSB ([chapter 4.2.6 "VSx and VDP" on page 38](#))
 - IndraControl VPB ([chapter 4.2.7 "VPx and VDP" on page 41](#))
 - IndraControl VDP ([chapter "VDP 08.3/16.3/40.3" on page 39](#) and [chapter "VDP 08.3/16.3/40.3" on page 42](#))
 - IndraControl VPP ([chapter 4.2.8 "VPP" on page 45](#))
 - IndraControl VSP ([chapter 4.2.9 "VSP" on page 48](#))

4.2.2 VCP

Brief Description

The VCP small operator panels are operating and visualization terminals for operating and observing machinery. The devices are suitable for a number of different applications thanks to their compact design. Communication with the superordinate control is either via PROFIBUS DP or Ethernet interface.

Overview



Fig.4-1: Small operator panel VCP 02



Fig.4-2: Small operator panel VCP 05

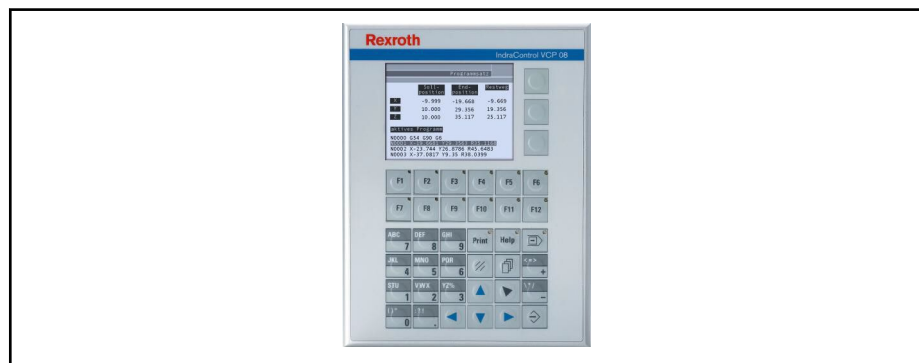


Fig.4-3: Small operator panel VCP 08

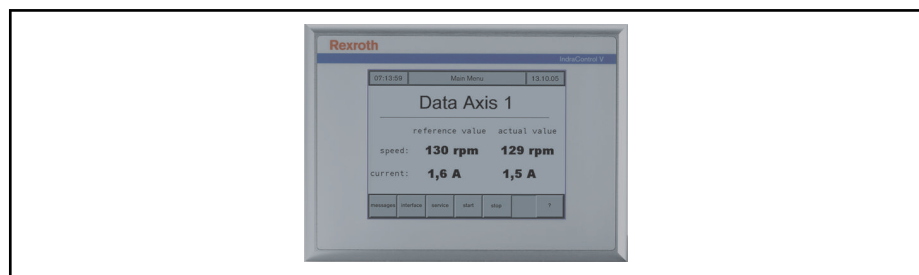


Fig.4-4: Small operator panel VCP 11

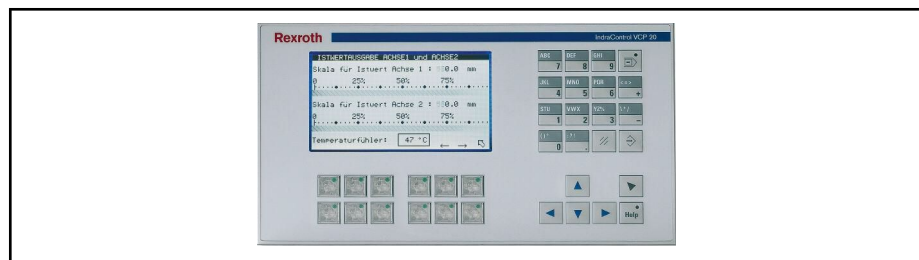


Fig.4-5: Small operator panel VCP 20



Fig. 4-6: Small operator panel VCP 25



Fig. 4-7: Small operator panel VCP 35

Technical Data

	VCP 02	VCP 05	VCP 08	VCP 11	VCP 20	VCP 25	VCP 35
Display	3" graphic display, 160 x 80 pixels			3.8" graphic display, 320 x 240 pixels	5,7" graphic display, 320 x 240 pixels	5,7" graphic display, 320 x 240 pixels	10.4" graphic display, 640 x 480 pixels
Front panel	100 x 148 mm	168 x 120 mm	159 x 209 mm	96 x 130 mm	160 x 300 mm	180 x 234	249 x 328 mm
Function/system buttons	4 / 7	6 / 24		Touch screen	16 / 22	Touch screen	Touch screen
Processor	ARM 200 MHz						
Application memory	3 MB						
Interfaces	1x Ethernet (10/100 Base T) 2x USB host 2.0 Profibus DP interface						
Power supply	24 V DC						

Fig. 4-8: Technical data for VCP small operating terminal

Models

The following models of VCP small operator panels are available:

Type	Remarks
VCP02.2DRN-003-NN-NN-PW	Pushbutton, communication via Ethernet
VCP02.2DRN-003-PB-NN-PW	Pushbutton, communication via PROFIBUS DP

Overview

Type	Remarks
VCP05.2DSN-003-NN-NN-PW	Pushbutton, communication via Ethernet
VCP05.2DSN-003-PB-NN-PW	Pushbutton, communication via PROFIBUS DP
VCP08.2DTN-003-NN-NN-PW	Pushbutton, communication via Ethernet
VCP08.2DTN-003-PB-NN-PW	Pushbutton, communication via PROFIBUS DP
VCP11.2DWN-003-NN-NN-PW	Touch screen, communication via Ethernet
VCP11.2DWN-003-PB-NN-PW	Touch screen, communication via PROFIBUS DP
VCP20.2DUN-003-NN-NN-PW	Pushbutton, communication via Ethernet
VCP20.2DUN-003-PB-NN-PW	Pushbutton, communication via PROFIBUS DP
VCP25.2DVN-003-NN-NN-PW	Touch screen, communication via Ethernet
VCP25.2DVN-003-PB-NN-PW	Touch screen, communication via PROFIBUS DP
VCP35.2ECN-003-NN-NN-PW	Touch screen, communication via Ethernet
VCP35.2ECN-003-PB-NN-PW	Touch screen, communication via PROFIBUS DP

Fig.4-9: VCP device models

Accessories

Connection cable

The VCPs communicate with the controls either using PROFIBUS DP or Ethernet. The following cable is available for PROFIBUS DP:

Type	Remark
IKB0034/000,0	Connection between control and VCP; cable assembly on one side, variable length

Fig.4-10: PROFIBUS connection cable

The following cables are available for communication via Ethernet:

Type	Remark
RKB0008/002.5	Bus cable, Ethernet cable assembly, 100-Base-T, CAT 6+ assembly, UL, drag chain-compatible, cable length 2.5 m
RKB0008/005,0	as above, cable length 5 m
RKB0008/010,0	as above, cable length 10 m
RKB0008/015,0	as above, cable length 15 m
RKB0008/020,0	as above, cable length 20 m

Type	Remark
RKB0008/025,0	as above, cable length 25 m
RKB0008/030,0	as above, cable length 30 m
RKB0008/035,0	as above, cable length 35 m
RKB0008/040,0	as above, cable length 40 m
RKB0008/050,0	as above, cable length 50 m

Fig. 4-11: Ethernet connection cable

Battery kit

A battery kit is available under the following designation:

Type	Remark
VAS04.1-001-002-NN	Battery kit

Fig. 4-12: Battery kit for VCP devices

Documentation

The following documentation is available to provide comprehensive descriptions of the VCP small operator panels:

- DOK-SUPPL*-VCP*02.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*05.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*08.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*11.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*20.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*25.2***-PRxx-EN-P
- DOK-SUPPL*-VCP*35.2***-PRxx-EN-P

xx - Edition

4.2.3 VCH

Brief Description

Operating, installing, parameterizing and diagnosing from anywhere – IndraControl VCH 08 makes it possible using Ethernet TCP/IP. When connecting and unplugging, the Stop function is automatically bridged and guarantees trouble-free operation.

Low weight and optimum design allow users to work without fatigue and provide excellent operating comfort.

The integral 3-stage enabling button and 2-circuit design of Stop button are looped through the VAC 30 switch-on box and thus ensure maximum safety.

Special features:

- Safety functions for operators and machinery
- Handles with integral enabling and stop buttons
- Sturdy construction for industrial use
- Ergonomically optimized for safe handling without the risk of fatigue
- Brilliant 3.8" graphic display and an all-purpose usable foil keyboard for convenient operation and visualization

Overview



Fig.4-13: IndraControl VCH 08

Field of Application

The VCH 08 is designed for use in handling applications.

Technical Data

	VCH 08
Display	3.8" Grayscale, 320 x 240 pixels
Front panel design	15 function keys 25 system keys 40 operating keys
Enable key	Number: 2 2-circuit, 3-stage
Stop button	Number: 1 2-circuit
Handwheel	Number: 1
Override potentiometer	Number: 1
Processor	CPU PXA 270/416 MHZ (Standard
memory	64 MB (application) 64 MB (flash)

Fig.4-14: Technical data for VCH 08

Models

The following models of VCH 08 are available:

Type	Remarks
VCH08.1EAB-064ET-A1D-064-CS-E2-PW	with handwheel (0.5 m Ethernet control cable)
VCH08.1EAB-064ET-A1D-064-CS-E1-PW	(8 m Ethernet control cable)
VCH08.1EAB-064ET-A1D-064-DS-E1-PW	with handwheel (8 m Ethernet control cable)

Fig.4-15: VCH 08 models

Accessories

The following accessories are available for the VCH 08:

Type	Remarks
VAS01.1-002-NNN-NN	Wall bracket VAS 01
VAC30.2N-NN	VAC 30

Fig.4-16: Accessories for the VCH 08

Documentation

The following documentation is available for the comprehensive description of the VCH 08 Hand-held terminal:

- DOK-SUPPL *-VCH*08.1***-PRxx-EN-P
xx - Edition

4.2.4 VEP
Brief Description

These terminals allow users to operate their machine easily and conveniently using a touch screen or a virtual keyboard. These devices, similar to PCs, only use embedded components for a robust construction.

Thanks to the large number of interfaces and slots, the IndraControl VEP devices can be ideally adapted to comply with machine and system requirements. This flexibility allows you to use the devices purely for visualization or as control hardware.

Special features:

- Compact device design for installation in control cabinets or on a support arm.
- Non-hard drive hardware without rotating media
- Control and visualization in one device
- Visualization, operation and observation with WinStudio visualization software throughout



Fig.4-17: IndraControl VEP variants

Brief Description

The Standard Industrial Operating Terminals VEP 30.3/40.3/50.3 are primarily used for all tasks in PC-based automation.

Overview

Technical Data

	VEP 30.3	VEP 40.3	VEP 50.3
Display	8.4" color display	12" color display	15" color display
Keyboard	via virtual keyboard		
Module slots	1 slot		
Interfaces	2 x USB 2.0 - ports (type A) 2 x Ethernet connection (RJ 45, 10/100 Base-T) 1 x serial standard interface RS232 (9-pin; D-Sub)		
Field bus	Field bus module (PROFIBUS DP-Master) is assigned 1 slot		
Processor	Celeron 600 MHz		
Primary memory	512 MB		
Compact Flash	Equipped with 2 CF socket/ 1GB CF-card		
Supply voltage	24 V DC		
Operating system	Windows CE 4.2		

Fig. 4-18: Technical data for VEP 30.3/40.3/50.3

	VEP 30.4	VEP 40.4	VEP 50.4
Display	8.4" color display	12" color display	15" color display
Keyboard	Touch control	Touch control or touch and key control	Touch control or touch and key control
Module slots	--		
Interfaces	3 x USB 2.0 - Connection (type A) 1 x Ethernet port (RJ 45, 10/100 Base-T)		
Processor	Intel Atom processor 1,1 GHz		
Primary memory	1 GB		
Compact Flash	2 CF socket / Equipped with 1GB CF-card or 4 GB CF-card		
Supply voltage	24 V DC		
Operating system	Windows CE 6.0 / Windows XPe		

Fig. 4-19: Technical data for VEP 30.4/40.4/50.4

Wear Parts

Parts are installed in the Standard Industrial Terminals VEP 30.3/40.3/50.3 that have a limited service life and are therefore not covered by warranty. This affects the following components:

- CF card
- CMOS battery
- Background lighting

The service life of the individual components is described in the device documentation (see [Chapter Documentation, page 36](#)).

Models

The following models of standard industrial operating terminals VEP 30.3/40.3/50.3 are available:

Overview

Type	Slots	Remarks
VEP30.3CCN-256NA-MAD-128-NN-FW	1	Touch screen (suitable for use in the food industry)
VEP30.3DKU-256NA-MAD-128-CG-FW	1	Touch screen, housed in an IP 54 aluminum casing, 3 buttons and E-Stop.
VEP40.3CEU-256NA-MAD-128-NN-FW	1	Touch screen (suitable for use in the food industry)
VEP50.3CHU-256NA-MAD-128-NN-FW	1	Touch screen (suitable for use in the food industry)

Fig. 4-20: VEP 30.3/40.3/50.3 models

The following models of standard industrial operating terminals VEP 30.4/40.4/50.4 are available:

Type	Slots	Remarks
VEP30.4EFN-512NN-A2D-NNN-NN-FW	-	Touch screen with front USB
VEP40.4DBN-512NN-A2D-NNN-NN-FW	-	Touch screen
VEP40.4BKN-512NN-A2D-NNN-NN-FW	-	Touch screen with front USB and machine function keys
VEP50.4DEN-512NN-A2D-NNN-NN-FW	-	Touch screen
VEP50.4BIN-512NN-A2D-NNN-NN-FW	-	Touch screen with front USB and machine function keys

Fig. 4-21: Technical data for VEP 30.4/40.4/50.4

Overview

Accessories

The following accessories are available for the VEP xx.3 devices:

Type	Remarks
VAS02.1-001-NNN-NN	Keyboard extension VAS 02

Fig.4-22: Accessories for the VEP 30.3/40.3/50.3

The following accessories are available for the VEP xx.4 devices:

Type	Remarks
IKB0033/000,0	Profibus cable with connectors at both ends
IKB0034/000,0	Profibus cable with one connector and open end
CFM01.1-01G0-N-LBA-NN-NW	Compact flash module 1 GByte
CFM01.1-04G0-N-LBA-NN-NW	Compact flash module 4 GByte

Fig.4-23: Accessories for VEP 30.4/40.4/50.4

Documentation

The following documentation is available for the detailed description of the Compact Industrial PCs VEP 30.3/40.3/50.3:

- DOK-SUPPL*-VEP**.*3****-PRxx-EN-P

xx - Edition

The following documentation is available for the detailed description of the Compact Industrial PCs VEP 30.4/40.4/50.4:

- DOK-SUPPL*-VEP**.*4****-PRxx-EN-P

xx - Edition

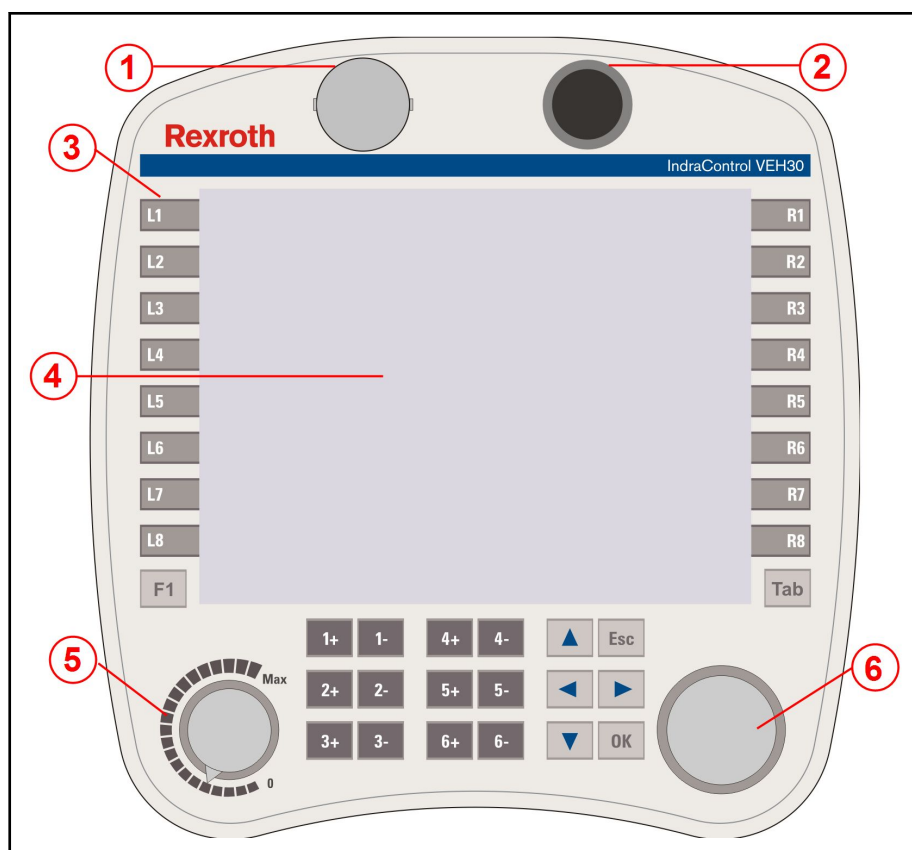
4.2.5 VEH

Brief Description

The hand-held terminal VEH 30.2 is a portable, PC-based operating and visualization device.

The two-circuit, three-stage enabling push button as well as a two-circuit STOP push button ensure a safe handling of IndraLogic and IndraMotion controls. VEH 30.2 facilitates the setting up of machines and systems that can be distributed across different connection points.

All device variants possess a 8,4" (213.36 mm) TFT display with eight function keys to the right and left of the monitor as well as jog keys for up to six axes for a user-friendly operation of the system. Besides, a USB interface on the front as well as an EtherNet interface on the VAC 30.2 connection module area available as service interface. Different cable variants permit customized working. In the following table, an overview of the most important characteristics of the VEH 30.2 is given. Differences between the device variants can be found under [chapter Models, page 38](#).



- ① USB connection
- ② STOP button
- ③ Touch panel
- ④ Display
- ⑤ Module slot for override
- ⑥ Mounting position for electronic handwheel

Fig. 4-24: Front view of VEH 30.2

Field of Application

The VEH 30.2 can be used for the following fields of application:

- Operating and visualizing IndraLogic XLC, IndraMotion MLC, IndraMotion MTX.
- Setting up machines and systems.
- Operating and the diagnostics of large handling systems.
- Remote visualization of control panels that have to be mounted.
- Visualizing basic operator control panels of mounting lines.

Technical Data

	VEH 30.2
Display	8.4" color display
Keyboard	Yes
Touch screen	Yes
Enabling push button	Two-circuit, three-stage
STOP button	Two-circuit

Overview

Interfaces	1 x Ethernet port (RJ 45, 10/100 Base-T)
Supply voltage	24 V DC

Fig.4-25: Technical data VEH 30.2

Wear Parts

Parts are installed in VEH 30.2 hand-held terminal that have a limited service life and do not fall under the warranty. This affects the following components:

- CMOS battery
- Background lighting

The service life of the individual components is described in the device documentation (see [Chapter Documentation, page 38](#)).

Models

The hand-held terminals VEH 30.2 are available as:

Device	Option elements, characteristics	Safety technology	Parts number
VEH30.2BNN-512ET-A2D-4G0-BS-E4-FW		STOP button, enabling push button	R911172223
VEH30.2BNN-512ET-A2D-4G0-DS-E4-FW	Hand-wheel, override	STOP button, enabling push button	R911171948
VEH30.2BNN-512ET-A2D-4G0-DS-E2-FW	Hand-wheel, override, short cable for extension	STOP button, enabling push button	R911172224
VEH30.2BNN-512ET-A2D-4G0-BS-E2-FW	Short cable for extension	STOP button, enabling push button	R911172162

Fig.4-26: Device variants VEH 30.2

Brief Description

The following documentation is available for the comprehensive description of the VEH 30.2 hand-held terminal:

- DOK-SUPPL*-VEH*30.2***-PRxx-EN-P

xx - Edition

4.2.6 VSx and VDP

Brief Description

The VSB 40.3 is an industrial PC, which together with the passive operating devices VDP 08.3, VDP 16.3 or VDP 40.3, represents a PC-based operating and visualization terminal with normal industrial capability. The VSB 40.3 is intended to be fitted in a control cabinet. The operating devices are designed for installation on a control panel or in a wall-mounted control cabinet. The connection between VDP 08.3/16.3/40.3 and VSB 40.3 is realized via a CDI interface.

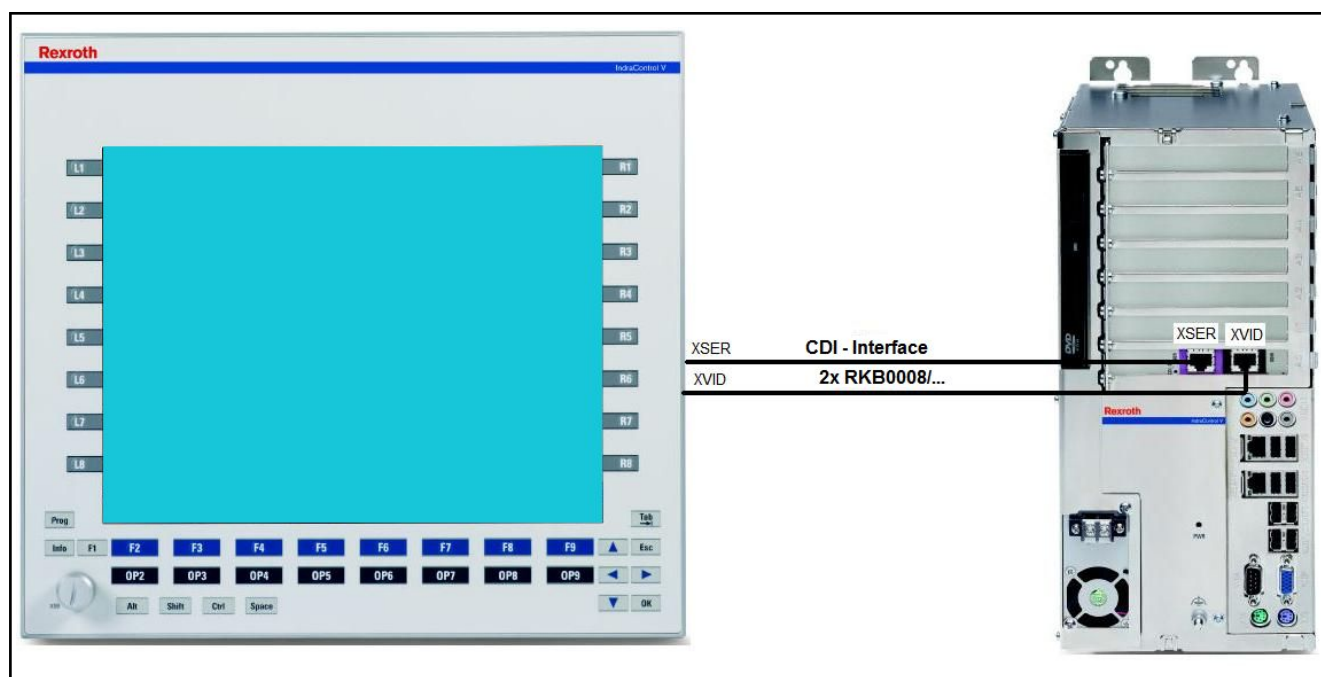


Fig.4-27: Standard industrial PC VSB 40.3 with operating device VDP 40.3

Field of Application

The Standard Industrial PCs VSB 40.3 are used for all tasks in PC-based automation.

Technical Data

VSB 40.3

	VSB 40.3
PC box	5x PCI / 1x PCIe
Processor	Celeron 440, min. 2 GHz or Intel Core 2™ E6400 2,13 GHz
Primary memory	1 GB RAM
Supply voltage	24V DC

Fig.4-28: Technical data for VSB 40.3

VDP 08.3/16.3/40.3

	VDP 08.3	VDP 16.3	VDP 40.3
Display	8" color display	12" color display	15" color display
Front panel design	16 machine function keys or touch screen		
Supply voltage	24 V DC		

Fig.4-29: Technical data for VDP 08.3/16.3/40.3

Wear Parts

The Standard Industrial PC VSB 40.3 and Operating Device VDP 08.3/16.3/40.4 have parts that have a limited service life and are therefore not covered by warranty.

This affects the following components:

- Hard drive

Overview

- CMOS battery
- Fan
- Background lighting

The service life of the individual components is described in the device documentation (see [Chapter Documentation, page 41](#)).

Models

The following models of Standard Industrial PC VSB 40.3 are available:

Type	Slots	Remarks
VSB40.3D1G-1G0NN-C2D-DN-NN-FW	6	without DVD drive
VSB40.3D1G-1G0NN-C2D-DE-NN-FW	6	with DVD drive
VSB40.3D1G-1G0NN-C2D-EN-NN-FW	6	Without DVD drive, 2x 80 GB hard drive
VSB40.3D1G-1G0NN-C2D-EE-NN-FW	6	With DVD drive; 2x 80 GB hard drive
VSB40.3D1G-1G0NN-C2D-FN-NN-FW	6	without DVD drive; 1 x SSD hard drive
VSB40.3D1G-1G0NN-C2D-FE-NN-FW	6	with DVD drive; 1 x SSD hard drive

Fig.4-30: Selection for VSB 40.3

The following models of VDP 08.3/16.3/40.3 operating devices are available:

Type	Remarks
VDP08.3DPN-D1-NN-CG	Operator terminal with 8.4" display, touch screen, in a housing with switching elements
VDP16.3DBN-D1-NN-NN	Operating device with 12" color display and touch screen with key panel
VDP40.3DIN-D1-NN-CG	Operator terminal with 15" display, touch screen, in a housing with switching elements
VDP40.3DEN-D1-NN-NN	Operating device with 15" color display and touch screen

Fig.4-31: Selection for VDP 08.3/16.3/40.3

Accessories

Connecting cable (CDI interface)

The industrial PCs VPB 40.3 and the operating device VDP 08.3/16.3/40.3 are connected via the CDI interface. The following cable assemblies are available:

Type	Remark
RKB0008/002,5	Bus cable, Ethernet cable assembly, 100-Base-T, CAT 6+ assembly, UL, drag chain-compatible, cable length 2.5 m
RKB0008/005,0	as above, cable length 5 m
RKB0008/010,0	as above, cable length 10 m
RKB0008/015,0	as above, cable length 15 m
RKB0008/020,0	as above, cable length 20 m
RKB0008/025,0	as above, cable length 25 m
RKB0008/030,0	as above, cable length 30 m
RKB0008/035,0	as above, cable length 35 m
RKB0008/040,0	as above, cable length 40 m
RKB0008/050,0	as above, cable length 50 m

Fig. 4-32: Connecting cable VSB 40.3 - VDP 8.3/16.3/40.3



2 connecting cables type RKB0008/... are required to connect a VSB 40.3 and a VDP 08.3/16.3/40.3.

Accessories for VSB 40.3

Type	Remark
VAP01.1H-W23-024-010-NN	Power supply unit 24 V DC; 10A
VAU01.1U-024-024-240-NN	Uninterruptible power supply with 24 V input/24 V output/240 W
RKB0019/...	USB interface cable for connection of the interruption-free power supply; lengths: 0,5 m; 1,0 m; 3,0 m; 5,0 m

Fig. 4-33: Accessories for VSB 40.3

Documentation

The following documentation is available for a detailed description of the Standard Industrial PC VSB 40:

- DOK-SUPPL*-VSB*40.3***-PRxx-EN-P
- DOK-SUPPL*-VAU*01.1U**-PRxx-EN-P

The following documentation is available for a detailed description of the operating devices VDP 08.3/16.3/40.3:

- DOK-SUPPL*-VDP*XX.3***-PRxx-EN-P

xx - Edition

4.2.7 VPx and VDP

Brief Description

The VPB 40.3 is a high-end industrial PC, which together with the passive VDP 08.3, VDP 16.3 or VDP 40.3 operating devices, represents a PC-based operating and visualization terminal with excellent industrial capability. The VPB 40.3 is intended to be fitted in a control cabinet. The operating devices are designed for installation on a control panel or in a wall-mounted control

Overview

cabinet. The connection between VDP 08.3/16.3/40.3 and VPB 40.3 is via a CDI interface.

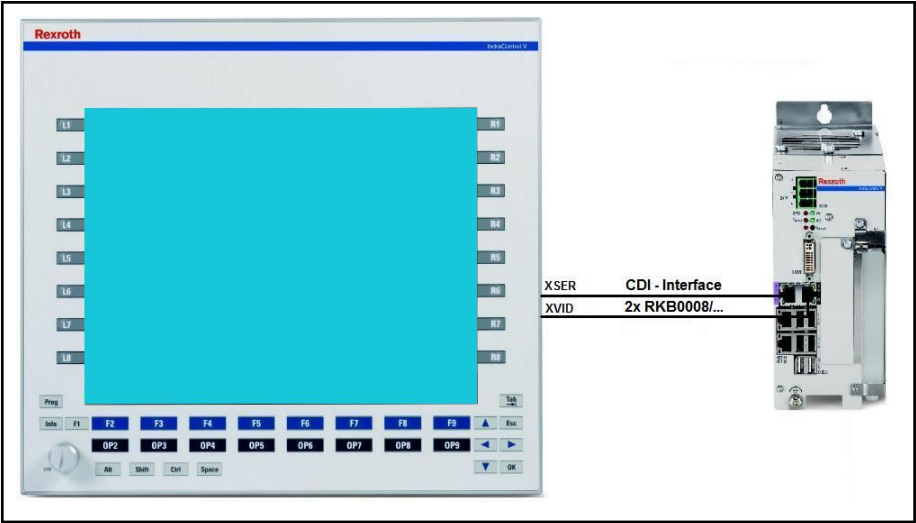


Fig.4-34: High end industrial PC VPB 40.3 with operating device VDP 40.3

Field of Application

The Compact Industrial PC VPB 40.3 is used in industrial environments which demand higher vibration and shock requirements.

Technical Data

VPB 40.3

Description	VPP40.3
PC box	1, 2 or 4 slots
Processor	Intel Core™ Duo L2400 1.66 GHz or Intel Core 2™ Duo T7400 2.16 GHz
Primary memory	1 GB, 2 GB or 4 GB RAM
Supply voltage	24 V DC

Fig.4-35: Technical data for VPB 40.3

VDP 08.3/16.3/40.3

	VDP 08.3	VDP 16.3	VDP 40.3
Display	8" color display	12" color display	15" color display
Front panel design	16 machine function keys or touch screen		
Supply voltage	24 V DC		

Fig.4-36: Technical data for VDP 08.3/16.3/40.3

Wear Parts

The Compact Industrial PC VPB 40.3 and Operating Device VDP 08.3/16.3/40.4 have parts that have a limited service life and are therefore not covered by warranty.

This affects the following components:

- Hard drive

- CMOS battery
- Fan
- Background lighting

The service life of the individual components is described in the device documentation (see [Chapter Documentation, page 45](#)).

Models

The following models of high-end industrial PCs VPB 40.3 are available:

Type	Slots	Remarks
VPB40.3D1K-1G0NN-D1D-DN-NN-FW	1	without DVD drive
VPB40.3D1K-1G0NN-D1D-FN-NN-FW	1	without DVD drive; SSD hard drive
VPB40.3D1K-2G0NN-D2D-EN-NN-FW	1	Core 2 Duo processor; without DVD drive; 2x 80 GB hard drive; 2 GB RAM memory
VPB40.3D1L-1G0NN-D1D-DE-NN-FW	2	with DVD drive
VPB40.3D1L-1G0NN-D1D-DN-NN-FW	2	without DVD drive
VPB40.3D1L-1G0NN-D1D-EE-NN-FW	2	With DVD drive; 2x 80 GB hard drive
VPB40.3D1L-1G0NN-D1D-EN-NN-FW	2	Without DVD drive; 2x 80 GB hard drive
VPB40.3D1L-1G0NN-D1D-FN-NN-FW	2	without DVD drive; SSD hard drive
VPB40.3D1L-2G0NN-D2D-DN-NN-FW	2	Core 2 Duo processor; without DVD drive; 2 GB RAM memory
VPB40.3D1L-2G0NN-D2D-EE-NN-FW	2	Core 2 Duo processor; with DVD drive; 2x 80 GB hard drives 2 GB RAM memory
VPB40.3D1L-2G0NN-D2D-EN-NN-FW	2	Core 2 Duo processor; without DVD drive; 2x 80 GB hard drive; 2 GB RAM memory
VPB40.3D1L-2G0NN-D2D-FN-NN-FW	2	Core 2 Duo processor; without DVD drive; SSD hard drive; 2 GB RAM memory
VPB40.3D1M-1G0NN-D1D-DE-NN-FW	4	with DVD drive
VPB40.3D1M-1G0NN-D1D-DN-NN-FW	4	without DVD drive
VPB40.3D1M-1G0NN-D1D-EE-NN-FW	4	With DVD drive; 2x 80 GB hard drive
VPB40.3D1M-1G0NN-D1D-FE-NN-FW	4	with DVD drive; SSD hard drive
VPB40.3D1M-1G0NN-D1D-FN-NN-FW	4	without DVD drive; SSD hard drive
VPB40.3D1M-2G0NN-D2D-DE-NN-FW	4	Core 2 Duo processor; with DVD drive; 2 GB RAM memory
VPB40.3D1M-2G0NN-D2D-DN-NN-FW	4	Core 2 Duo processor; without DVD drive; 2 GB RAM memory

Overview

Type	Slots	Remarks
VPB40.3D1M-2G0NN-D2D-EE-NN-FW	4	Core 2 Duo processor; with DVD drive; 2x 80 GB hard drives 2 GB RAM memory
VPB40.3D1M-2G0NN-D2D-FN-NN-FW	4	Core 2 Duo processor; without DVD drive; SSD hard drive; 2 GB RAM memory
VPB40.3D1M-4G0NN-D2D-EE-NN-FW	4	Core 2 Duo processor; with DVD drive; 2x 80 GB hard drives 2 GB RAM memory

Fig. 4-37: Selection for VPD 40.3

The following models of VDP 08.3/16.3/40.3 operating devices are available:

Type	Remarks
VDP08.3DPN-D1-NN-CG	Operator terminal with 8.4" display, touch screen, in a housing with switching elements
VDP16.3DBN-D1-NN-NN	Operating device with 12" color display and touch screen with key panel
VDP40.3DIN-D1-NN-CG	Operator terminal with 15" display, touch screen, in a housing with switching elements
VDP40.3DEN-D1-NN-NN	Operating device with 15" color display and touch screen

Fig. 4-38: Selection for VDP 08.3/16.3/40.3

Accessories

Connecting cable (CDI interface)

The Compact PC VPB 40.3 and the Operating Device VDP 08.3/16.3/40.3 are connected via the CDI interface. The following cable assemblies are available:

Type	Remark
RKB0008/002.5	Bus cable, Ethernet cable assembly, 100-Base-T, CAT 6+ assembly, UL, drag chain-compatible, cable length 2.5 m
RKB0008/005,0	as above, cable length 5 m
RKB0008/010,0	as above, cable length 10 m
RKB0008/015,0	as above, cable length 15 m
RKB0008/020,0	as above, cable length 20 m
RKB0008/025,0	as above, cable length 25 m
RKB0008/030,0	as above, cable length 30 m
RKB0008/035,0	as above, cable length 35 m

Type	Remark
RKB0008/040,0	as above, cable length 40 m
RKB0008/050,0	as above, cable length 50 m

Fig. 4-39: Connecting cable VPB 40.3 - VDP 08.3/16.3/40.3



2 connecting cables type RKB0008/... are required to connect a VPB 40.3 and a VDP 08.3/16.3/40.3.

Accessories for VPB 40.3

Type	Remark
VAP01.1H-W23-024-010-NN	Power supply unit 24 V DC; 10A
VAU01.1U-024-024-240-NN	Uninterruptible power supply with 24 V input/24 V output/240 W
RKB0019/...	USB interface cable for connection of the interruption-free power supply; lengths: 0.5m; 1.0m; 3.0m; 5.0m

Fig. 4-40: Accessories for VPB 40.3

Documentation

The following documentation is available for a detailed description of the high-end industrial PC VPB 40.3:

- DOK-SUPPL*-VPB*40.3***-PRxx-EN-P
- DOK-SUPPL*-VAU*01.1U**-PRxx-EN-P

The following documentation is available for a detailed description of the operating devices VDP 08.3/16.3/40.3:

- DOK-SUPPL*-VDP*XX.3***-PRxx-EN-P

xx - Edition

4.2.8 VPP

Brief Description

The VPP 16.3/40.3 operating devices are active PC-based operating and visualization terminals with excellent industrial capability. These devices are predominantly installed on a control panel or on the wall of a control cabinet.

Overview

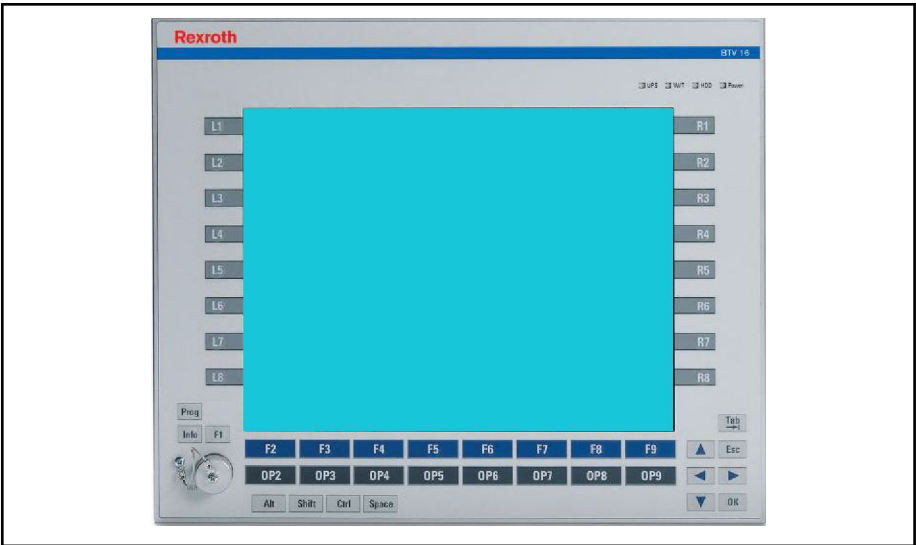


Fig.4-41: Compact Industrial PC VPP 16.3 with 12" color display

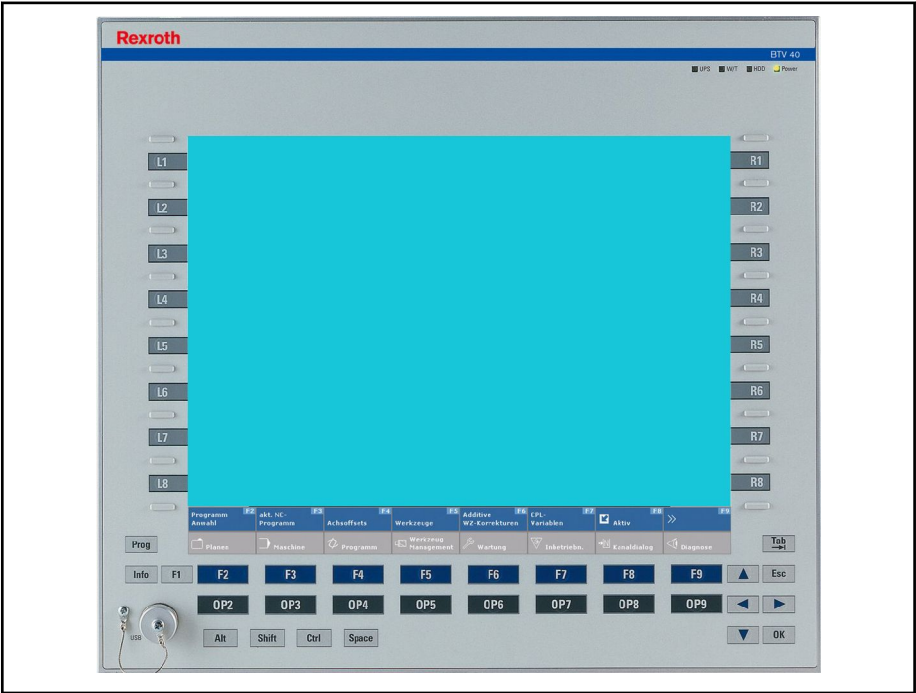


Fig.4-42: Compact Industrial PC VPP 40.3 with 15" color display

Field of Application

Compact Industrial PCs VPP 16.3/40.3/60.3 are used with standard applications right through to high-end applications.

Technical Data

	VPP 16.3	VPP 40.3	VPP 60.3
Display	12" color display	15" color display	17" color display
Front panel design	16 machine functional buttons		
PC box	1, 2 or 4 slots		

Processor	Intel Core™ Duo L2400 1.66 GHz or Intel Core 2™ Duo T7400 2.16 GHz
Primary memory	1 GB or 2 GB RAM
Supply voltage	24 V DC

Fig. 4-43: Technical data for VEP 16.3/40.3

Wear Parts

There are parts fitted to the Compact Industrial PC VPP 16.3/40.3 that have a limited service life and are therefore not covered by warranty.

This affects the following components:

- Hard drive
- CMOS battery
- Fan
- Background lighting

The service life of the individual components is described in the device documentation (see [Chapter Documentation, page 48](#)).

Models

The following models of Compact Industrial PCs VPP 16.3/40.3 are available:

Type	Slots	Remarks
VPP16.3DBK-1G0NN-D1D-DN-NN-FW	1	Touch screen
VPP16.3DBM-1G0NN-D1D-DN-NN-FW	4	Touch screen
VPP40.3DEK-1G0NN-D1D-DN-NN-FW	1	Touch screen
VPP40.3DEM-1G0NN-D1D-DN-NN-FW	4	Touch screen
VPP40.3DEM-1G0NN-D1D-DE-NN-FW	4	Touch screen; with DVD drive
VPP60.3DEK-1G0NN-D1D-DN-NN-FW	1	Touch screen
VPP60.3DEM-1G0NN-D1D-DN-NN-FW	4	Touch screen
VPP60.3DEM-1G0NN-D1D-DE-NN-FW	4	Touch screen; with DVD drive

Fig. 4-44: VPP 16.3/40.3 models

Accessories

The following accessories are available for VPP devices:

Overview

Type	Remarks
VAP01.1H-W23-024-010-NN	Power supply unit 24 V DC; 10 A
VAU01.1U-024-024-240-NN	Uninterruptible power supply with 24 V input/24 V output/240 W
RKB0019/...	USB interface cable for the connection of the interruption-free power supply; lengths: 0,5 m; 1,0 m; 3,0 m; 5,0 m

Fig.4-45: Accessories for the VPP 16.3/40.3

Documentation

The following documentation is available for the detailed description of the Compact Industrial PCs VPP 16.3/40.3/60.3:

- DOK-SUPPL*-VPP*XX.3***-PRxx-EN-P
- DOK-SUPPL*-VAU*01.1U**-PRxx-EN-P

xx - Edition

4.2.9 VSP

Brief Description

The Standard Industrial PCs VSP 16.3/40.3 are the economical alternatives for PC-based operating and visualization devices. These devices are predominantly installed on a control panel or on the wall of a control cabinet.

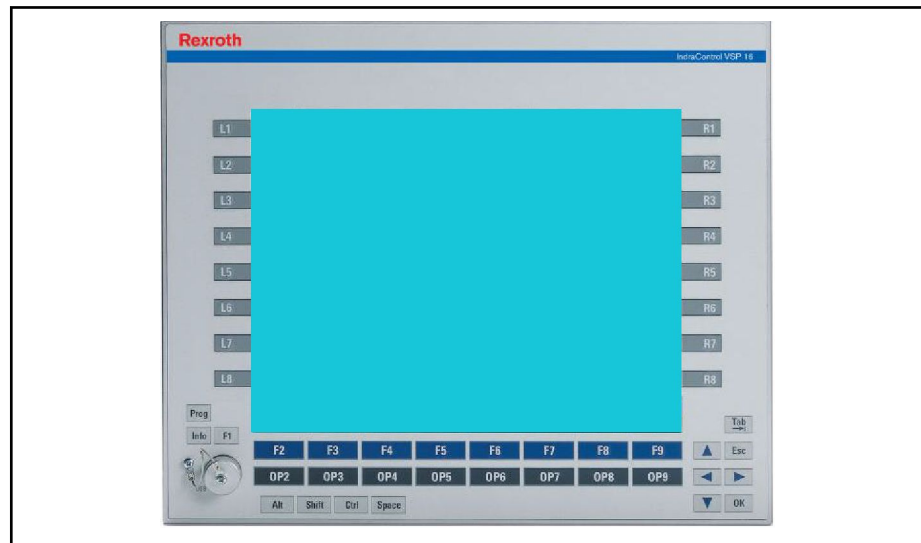


Fig.4-46: VSP 16.3 with 12" color display

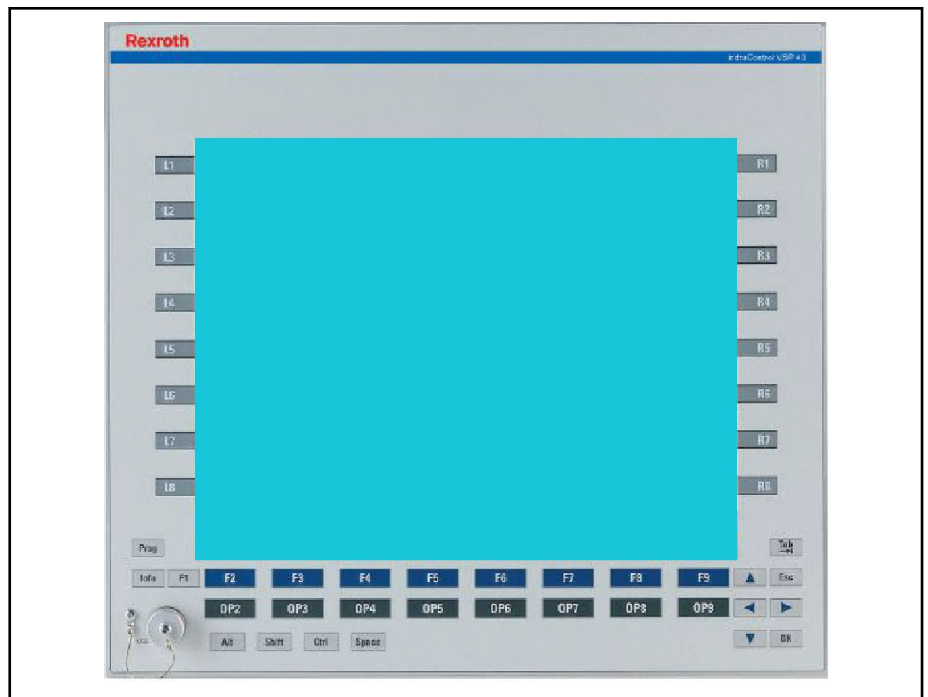


Fig.4-47: VSP 40.3 with 15" color display

Field of Application

The Standard Industrial PCs VSP are primarily used for all tasks in PC-based automation.

Technical Data

	VSP 16.3	VSP 40.3
Display	12" color display	15" color display
Front panel design	16 machine function keys or touch screen	
Module slots	5x PCI / 1x PCle	
Processor	Celeron 440, min. 2 GHz or Intel Core 2™ E6400 2,13 GHz	
Primary memory	1 GB RAM	
Supply voltage	24 V DC	

Fig.4-48: Technical data for VSP 16.3/40.3

Wear Parts

There are parts fitted to the Standard Industrial PCs VSP 16.3/40.3 that have a limited service life and are therefore not covered by warranty.

This affects the following components:

- Hard drive
- CMOS battery
- Fan
- Background lighting

The service life of the individual components is described in the device documentation (see [Chapter Documentation, page 50](#)).

Overview

Models

The following models of Standard Industrial PCs VSP 16.3/40.3 are available:

Type	Slots	Remark
VSP16.3DBG-1G0NN-C2D-DN-NN-FW	6	Touch screen; without DVD drive
VSP16.3DBG-1G0NN-C2D-DE-NN-FW	6	Touch screen; with DVD drive
VSP40.3DEG-1G0NN-C2D-DN-NN-FW	6	Touch screen; without DVD drive
VSP40.3DEG-1G0NN-C2D-DE-NN-FW	6	Touch screen; with DVD drive
VSP40.3DEG-1G0NN-C2D-FN-NN-FW	6	Touch screen; without DVD drive, with 16 GB SSD hard drive

Fig. 4-49: Parts number VSP 16.3/40.3

Accessories

The following accessories are available for VSP devices:

Type	Remark
VAP01.1H-W23-024-010-NN	Power supply unit 24 V DC; 10 A
VAU01.1U-024-024-240-NN	Uninterruptible power supply with 24 V input/24 V output/240 W
RKB0019/...	USB interface cable for connection of the interruption-free power supply; lengths: 0,5 m; 1,0 m; 3,0 m; 5,0 m

Fig. 4-50: Accessories for the VSP 16.3/40.3

Documentation

The following documentation is available for the detailed description of the operating devices VSP 16.3/40.3:

- DOK-SUPPL*-VSP*XX.3***-PRxx-EN-P
- DOK-SUPPL*-VAU*01.1U**-PRxx-EN-P

xx - Edition

4.3 Visualizations

4.3.1 Overview

The IndraLogic XLC and IndraMotion MLC system provide several visualizations. Each visualization has its own application area. The following visualizations are supported:

- VI-Logic: This visualization is primarily intended for the development and commissioning phase. The visualization runs in the IndraWorks development environment. Text as well as graphic elements are supported.

See [chapter 4.3.2 VI-Logic, page 51](#).

- VI-Composer: This software is used to configure all small operating terminals of type VCP and VCH. The functional range is limited to text representation and simple graphic elements.

See [chapter 4.3.3 VI-Composer, page 53](#).

- WinStudio: This visualization software is used to configure all PC-based and embedded systems. Text as well as graphic elements are supported.

See [chapter 4.3.4 WinStudio, page 55](#).

4.3.2 VI-Logic

Field of Application

Integrated visualization for test / Commissioning / Diagnostics.



The IndraLogic commissioning visualization "VI-Logic visualization" is only intended to support the commissioning of machinery.



The IndraLogic commissioning visualization "VI-Logic visualization" can run on the PC as well as on the control. Different features are integrated in the visualization.

Properties of the Integrated Visualization

- Within a visualization, any desired expressions, even function calls, are permitted.
- A visualization can be created as an object below an application (or in the "General module" folder).
- In contrast to IndraLogic 2.x, all of the visualization elements as well as the visualization objects themselves are implemented as IEC61131-3 function blocks. In this way, visualizations can be instanced in other visualizations, i.e. they can be referenced and used to extend other function blocks.
- A placeholder concept ensures that placeholders are treated as input variables of a visualization.
- Interfaces can be edited in an interface editor. The use of frame elements makes it easy to switch from a display of one visualization to another within a visualization object.
- Visualizations in the Project Explorer are managed by a Visualization Manager for each application, which defines common settings for all application-specific visualizations.
- A few general settings for all visualizations in one project (file paths, size adjustment, etc.) can be made in the visualization options.
- Each individual visualization has properties, such as its planned use (as "visualization", "Numpad/Keypad" or "dialog") or display size.
- The visualization editor also includes the ToolBox tab which includes the available visualization elements, and a properties editor for configuring the elements that are added.
- It is easy to arrange and group the visualization elements. The standard elements are provided using the visualization libraries in the project. IndraLogic 1.x visualizations can be imported.
- Visualizations can be animated using IndraLogic project variables that are entered directly or using expressions, i.e. combinations of the variables with operators and constants. This allows variable values to be

Overview

scaled, for example, so that the variable values are suitable for use in the visualization.

- Language switching (ANSI or UNICODE) in a visualization can be executed by using text lists.
- The variable values of an application can be modified and displayed using a text field element. The formatting for these inputs and output is based on the standard function `sprintf` (C library).
- External data sources can also be used.
- Each visualization element can be assigned a tool tip text.
- In addition to the zoom function, automatic scaling is also possible (the visualization adjusts to the screen size).
- Visualizations can be saved in libraries so they can be made available in other projects.
- Multiple language capability

The VI-Logic visualization supports the multiple language capability of static texts via "GlobaleTextLists" dynamic texts via "TextLists". Where dynamic texts are concerned, the text displayed varies depending on a variable value.

Visualization Objects

Commissioning visualizations can be created by means of various visualization objects which are divided in "Basic", "General Control Elements" "Measuring Devices" and "Special Control Elements" objects. The list below contains the available visualization objects:

Visualization Objects

Basic	General Control Elements	Measuring Devices	Special Control Elements
Pointer	Button	Pointer instrument	Trace
Rectangle	Table	Histogram	ActiveX element
Rounded rectangle	Text field	Bar graph	
Ellipse	Scrollbar		
Line			
Polygon			
Polyline			
Bézier curve			
View			
Frame			

Fig. 4-51: Visualization objects according to fields

The visualization objects are configured via static and dynamic properties as well as event handling for user inputs in the "Properties" window. Predefined actions, such as executing ST code (only assignments but no loops or branches are permitted), changing variables, converting screens and languages, internal commands, as well as opening and closing of dialogs, can be selected from the actions provided for user entries.

Supported Visualization Devices

No additional hardware is required for the visualization as visualization and engineering run on one device.

Engineering is facilitated as the configuration of the visualization and the PLC application on an interface.

Protocol

The integrated visualization can directly access the variables from the control - externally generated lists of variables or symbols are not required.

Automatic configuration of the communication - an error-prone communication configuration is not required.



The "VI-Logic" visualization and its visualization objects provides all variables for project planning, which are used in the PLC program (implementation) or which were declared with the attribute "{attribute 'linkalways'}".

This means that variables which are only contained in the declaration are optimized by the compiler from runtime code, for which reason they cannot be visualized or changed.

Documentation

A detailed description of the integrated visualization software is available in the IndraWorks online help.

4.3.3 VI Composer

Field of Application

This software VI-Composer is used to configure all small operating terminals of type VCP and VCH.



The configuration software VI Composer is part of IndraWorks and has to be ordered separately.

Configurable Small Operating Terminals

The following small operating terminals can be configured with the VI-Composer configuration software:

- VCP:
 - IndraControl VCP 02
 - IndraControl VCP 05
 - IndraControl VCP 08
 - IndraControl VCP 11
 - IndraControl VCP 20
 - IndraControl VCP 25
 - IndraControl VCP 35
- VCH
 - IndraControl VCH 08

See also chapter "HA_3_2_2 VCP"

Communication Protocol BRC-Symbolic

Brief Description

The protocol offers optional read and write access to all global data objects of the control.

The configuration software applies the data objects of the symbol file (file-name.SYM or filename.xml) that is generated during the compilation of the control object.

The connected operating device access a file object via the symbolic name.

Data Objects

Generally, the variable length is specified by a definition in the "IndraLogic" configuration software.

Individual Variables

Variables of type BOOL, SINT, INT, DINT, BYTE, USINT, WORD, UINT, LWORD, DWORD, UDINT, REAL, STRING, WSTRING, LINT, ULINT, LREAL, BITORBYTE, ENUM, TIME, TOD, DATE and DT can be accessed.

String Variables

Use variable type STRING(N) or WSTRING(N) for string variables. N specifies the string length.

 String variables cannot exceed a length of 64 characters.

Tables

To display ARRAY variables in table fields, use data type ARRAY [1..N]. The ARRAY [1..N] has to consist of one of the following basic types:
BOOL, BYTE, WORD, LWORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, LINT, ULINT, LREAL, REAL, STRING, WSTRING, ENUM

Connection Settings

Protocol Parameter for PROFIBUS-DP

- Device address

This parameter specifies the operating device address.

Adjustable values	Default value
1 to 126	2

Fig.4-52: Device address

- Delay until connection is established.

This parameter specifies a waiting time until the communication is started by the operating device.

Adjustable values	Default value
5s to 255s	5s

Fig.4-53: Delay time when establishing a connection

- Reaction time slave

This parameter specifies how long the operating device waits for control response

Adjustable values	Default value
10ms to 65535ms	1000ms

Fig. 4-54: Reaction time slave

Protocol parameter for EtherNet

Connection table:

Connections to a maximum of 16 controls can be defined in a connection table.

The connection names of the main dialog are applied in the connection name column.

Enter a valid IP address for each connection name in the IP address column of the control.

Documentation

The following documentation is available to provide comprehensive descriptions of the VI-Composer configuration software:

DOK-SUPPL*-VIC*BED*0... en-EN

4.3.4 WinStudio

Overview

WinStudio is a visualization software for all PC-based and embedded systems.

The product consists of two parts.

WinStudio Engineering

- Configuration tool to create individual HMI screens up to complete operator interfaces.
- One-time investment (license) for any number of WinStudio projects.
- Scaling
 - WinStudio Lite license (limited functions; contained in the standard scope of the IW-systems).
 - Scaled license model according to number of used tags (process variables).
- WinStudio Engineering package characteristics.
 - Integral component part in IndraWorks Engineering to create visu objects.
 - Addition to IndraWorks HMI interface (IndraWorks OPD).
 - Creating individual visualizations (WinStudio Runtime stand-alone).
 - Stand-alone editor to create individual visualizations (WinStudio Runtime stand-alone).

WinStudio Runtime

- Software on visualization devices
 - pre-installed or
 - prepared for installation (software download).
- Scaling

Overview

- WinStudio Lite license (limited functions; contained in the standard scope of the IW-systems).
- Scaled license model according to number of used tags (process variables).
- WinStudio runtime environment characteristics.
 - WinStudio in IndraWorks HMI interface (IndraWorks OPD).
(User interface of the MLC, MTX, IL systems under Windows CE, XP)
 - WinStudio Runtime stand-alone (IndraWorks independent user interface).

WinStudio Elements in the Project Explorer

To create WinStudio screens and to configure extended function (e.g. processes, alarms) and the communication connection, different editors and dialogs are available. They can be opened via the context menu of the "WinStudio" or other subfolders.

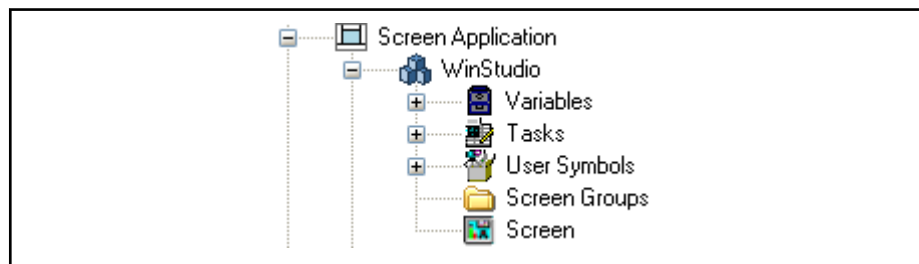


Fig.4-55: WinStudio folder

The following subfolder can be found below the WinStudio node in the Project Explorer:

- **Variables**
Editors for different variable types (application variables, system variables) can be found in this folder. Editors to configure the process communication (driver sheets) and to create individual data types (classes) are contained in this folder.
- **Tasks**
Extended WinStudio objects (e.g. mathematics, alarms) can be created and edited.
- **User symbols**
WinStudio objects created by the user that can be used for WinStudio screens are contained in this folder.
- **Screen groups**
Screen groups that summarize several WinStudio screens to a common screen can be created in this folder.

The following WinStudio elements are available if at least one license "WinStudio-Engineering and Runtime 1.5k" is activated and "WinStudio 1.5k" or higher is selected as target system of the WinStudio project.

- [chapter Alarms, page 71](#)
- [chapter Recipes, page 72](#)
- [chapter Reports, page 72](#)
- [chapter Trends, page 73](#)
- TCP/IP driver sheets

- DDE driver sheets

When opening a project in which "WinStudio 1.5k" or higher is selected as target system but no matching WinStudio license is active, the elements requiring a license are displayed. However, they are disabled and the user cannot edit them.

General WinStudio Settings and Functions

Project properties

Select "Project Settings" to open the dialog box project settings that can be used to check and change project settings to influence the entire application.

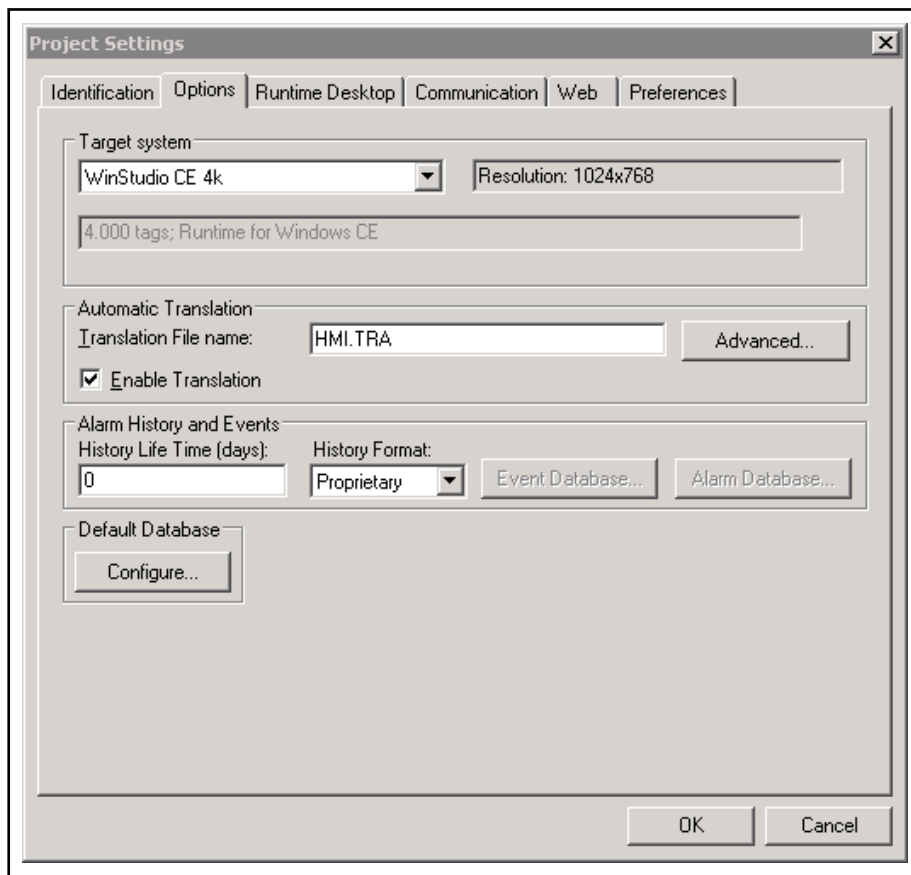


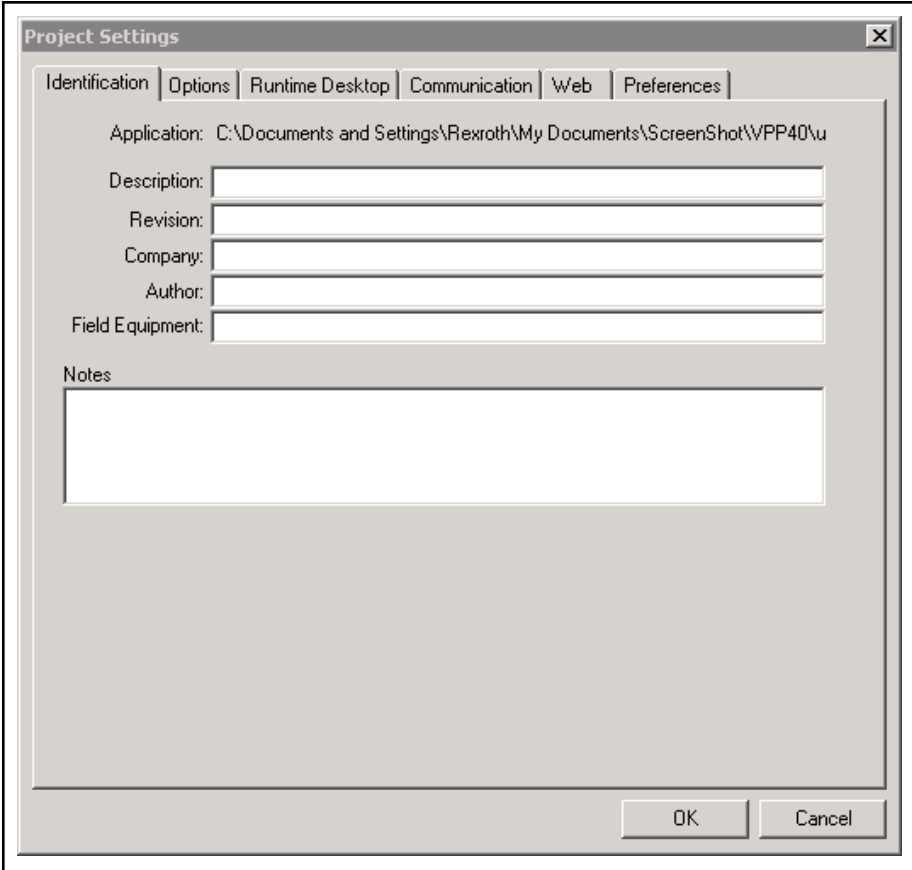
Fig.4-56: Dialog: Project settings

Subsequently, the tabs are written:

- Identification, page 58
- Options, page 59
- Runtime Desktop, page 60
- Communication, page 62
- Web, page 63

Overview

Identification



The screenshot shows a 'Project Settings' dialog box with a tabbed interface. The 'Identification' tab is selected. The dialog contains the following fields:

- Application: C:\Documents and Settings\Rexroth\My Documents\ScreenShot\WPP40\...
- Description: [Text input field]
- Revision: [Text input field]
- Company: [Text input field]
- Author: [Text input field]
- Field Equipment: [Text input field]
- Notes: [Large text area]

At the bottom right, there are 'OK' and 'Cancel' buttons.

Fig. 4-57: Dialog: Project settings (tab: Identification)

This tab in the Project Settings dialog serves to describe and identify the project for documentation purposes.

Options

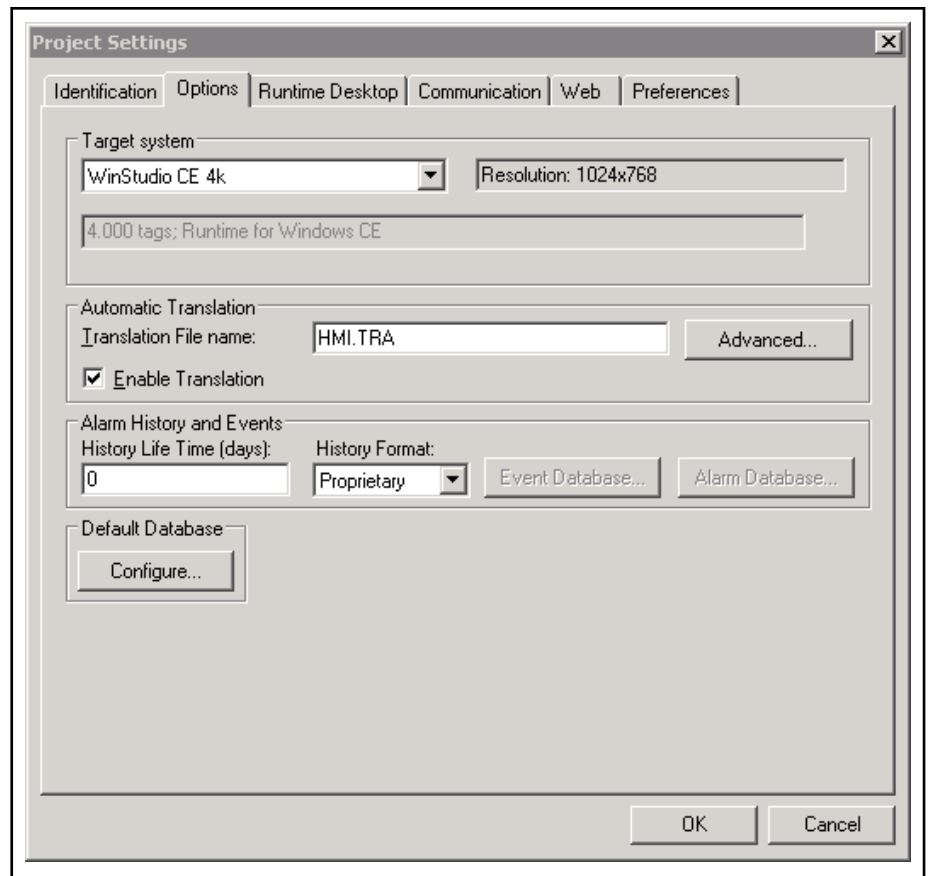


Fig. 4-58: Dialog: Project settings (tab: Options)

Use this tab to specify the parameters with regard to the application in general:

- **Target system**

By selecting the target system, the number of available WinStudio variables, the available functional scope as well as the project type matching for the target system (Windows XP/Vista or Windows CE) is specified. Default setting: "WinStudio Lite". More target systems with a higher number of variables and an extended functional scope are available after the WinStudio Engineering license has been activated.

The resolution is selected via the configured HMI device.



If a license is selected that does not match the current license on the operating station, the application can sporadically not be executed correctly.

- **Automatic translation**

Enable the checkbox "Enable translation" and enter a file name and the directory path to activate the automatic translation. WinStudio uses this translation file as default language upon the start of the application.

- **Alarm history + events**

Enter a value (days) in the "Delete after (days)" field to specify for which time duration alarm and event history files should be available. After this time has expired, WinStudio automatically deletes the available alarm

Overview

and event history files that are older than the specified time. If 0 is entered as value in this field, the history data are not automatically deleted. In this case, the user has to make sure that the files are deleted. Otherwise, the memory capacity is decreased considerably.

By selecting "Historic format", it can be specified how and where to save the alarm files.

Property	Saves history data in the subfolder \alarm of the application (by default) in text files by using the proprietary WinStudio format.
Database	Saves the history data in user-defined SQL relational databases by using the integrated ADO interface.
Binary	Saves history data in the subfolder \alarm of the application (by default) in binary coded files by using the proprietary WinStudio format.

- **Standard database**

A default database can be configured that can be used by different tasks and objects.

Runtime Desktop

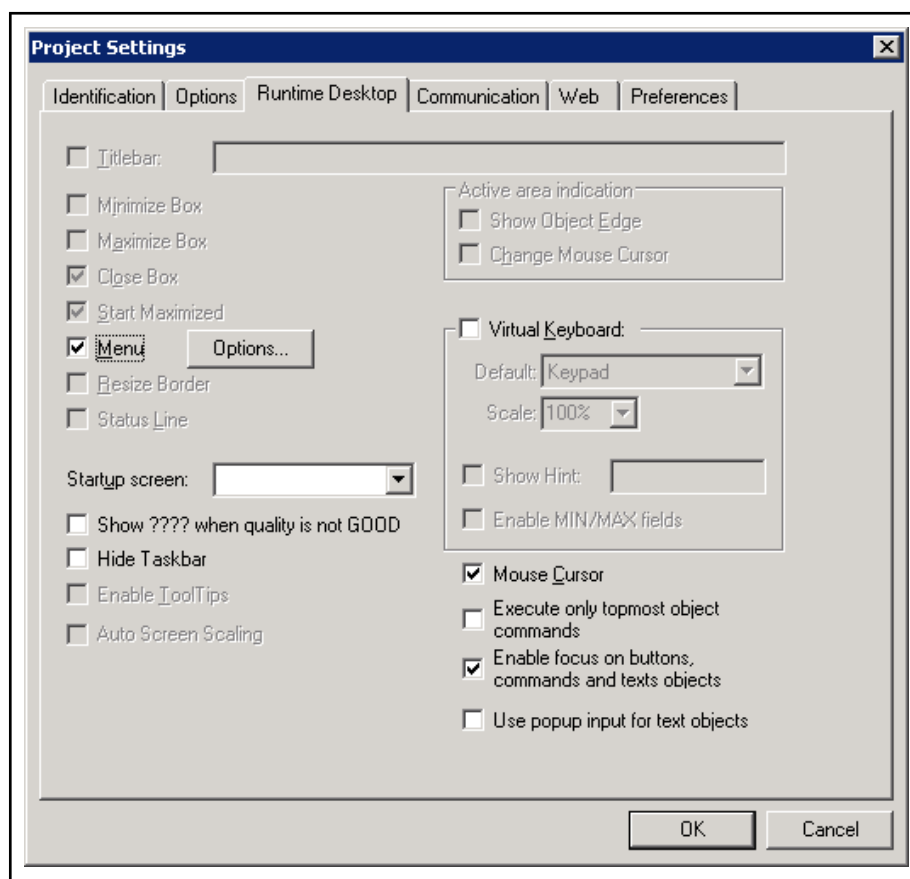


Fig. 4-59: Dialog: Project settings (tab: Runtime Desktop)

The Runtime Desktop parameter can be enabled/disabled in this tab:

- Check box and text field **Title**

Enable this box and enter a new name in the corresponding field to specify or change the default text for the title bar of the window.

- Check boxes **Minimize field, maximize field and close**

Enable these check boxes to show (enable) or hide (disable) these buttons in the window.

- Check box **Start full screen**
Enable this box to minimize the window automatically when the application is active.
- Check box **Menu and Options** interface
Enable the check box and the options interface to specify which menu options are available during runtime. In the "Runtime menu options" dialog, menu options can be shown (enable) or hidden (disable).

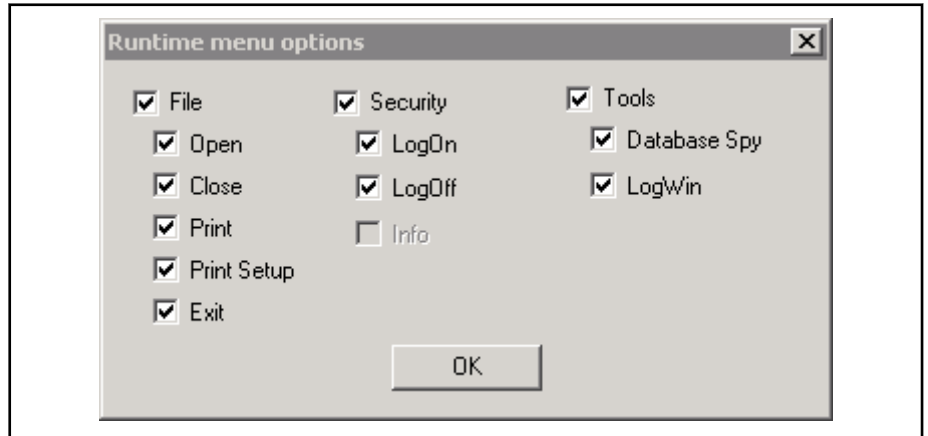


Fig.4-60: Dialog: Runtime menu options

- Check box: **Margin size**
Enable this box so that the user can change the window size during runtime.
- Check box: **Status bar**
Enable this box to show the status bar in the runtime application.
- Field: **Start Screen**
Enter the name of the screen that should be used as start screen when opening the runtime application.
- Check box: **Show ??? if quality is not good**
Enable this box to show (???) instead of the tag value if the quality is poor.
- Check box: **Hide tool bar**
Enable this box to hide the tool bar.
- Check box: **Allow ToolTips**
Enable this box to show Windows ToolTips if the application is running. ToolTips can be defined for each object in the Information field of the Object Properties dialog.
- Check box: **Auto screen scaling**
Enable this box to automatically adjust the size of the user screens when changing the window size. This function is available for local applications under Windows NT/2000/XP (or Web Thin Clients). This parameter is not available for local applications under Windows CE.
- Section: **Show active areas**
Enable the check boxes "Object margins" and "Change mouse pointer" in this area to change the object edge and the mouse cursor when the cursor hovers over an object.
- Section and check box: **Virtual keyboard**

Overview

Using this option, the virtual keyboard can be enabled/disabled. Data can be entered on the touch screen during runtime if the option is enabled.

- Check box: **Mouse pointer**

Enable this box to show the mouse cursor in the runtime application.

Communication

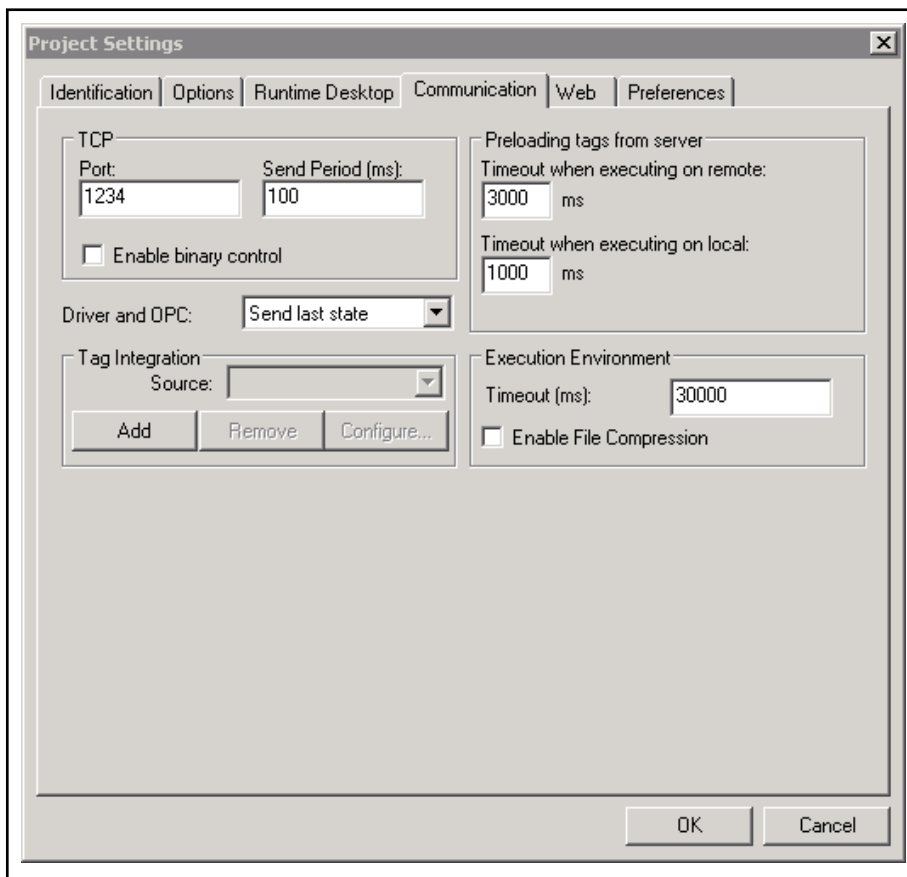


Fig.4-61: Dialog: Project settings (tab: Communication)

Use this tab to specify the communication parameters with regard to the application in general:

- **Driver and OPC**

Selecting the method that is used by all communication drivers and OPC client worksheets in the current application.

- **Send each status**

If this option is enabled, each value change of a tag is transmitted to the control.

- **Send last status**

If this option is selected, only the current (last) value is transmitted to the control. This is the desired behavior for most applications.

- **TCP**

Configuration dialog for TCP/IP Client and TCP/IP server module settings

- **Port**

TCP port that is used by the TCP/IP Client and by the TCP/IP server modules that run on the current computer. If this value is

changed in the local application, it is ensured that the same value is also changed in the remote application that communicates with the local application.

– **Transmission period (ms)**

Time (in milliseconds) between two messages that are send by the TCP/IP Client or the TCP/IP server modules to update tag values in the remote station. Generally, a smaller number stands for a better performance and more traffic in the network.

– **Allow binary control**

Mark this option to enable encryption if the TCP/IP Client or the TCP/IP server module sends messages to the remote station. Encryption increases the level of security of the system but lowers the communication performance. If two stations have been connected via the TCP/IP Client and the TCP/IP server module, please ensure that this setting is either enabled or disabled on both applications.

Web

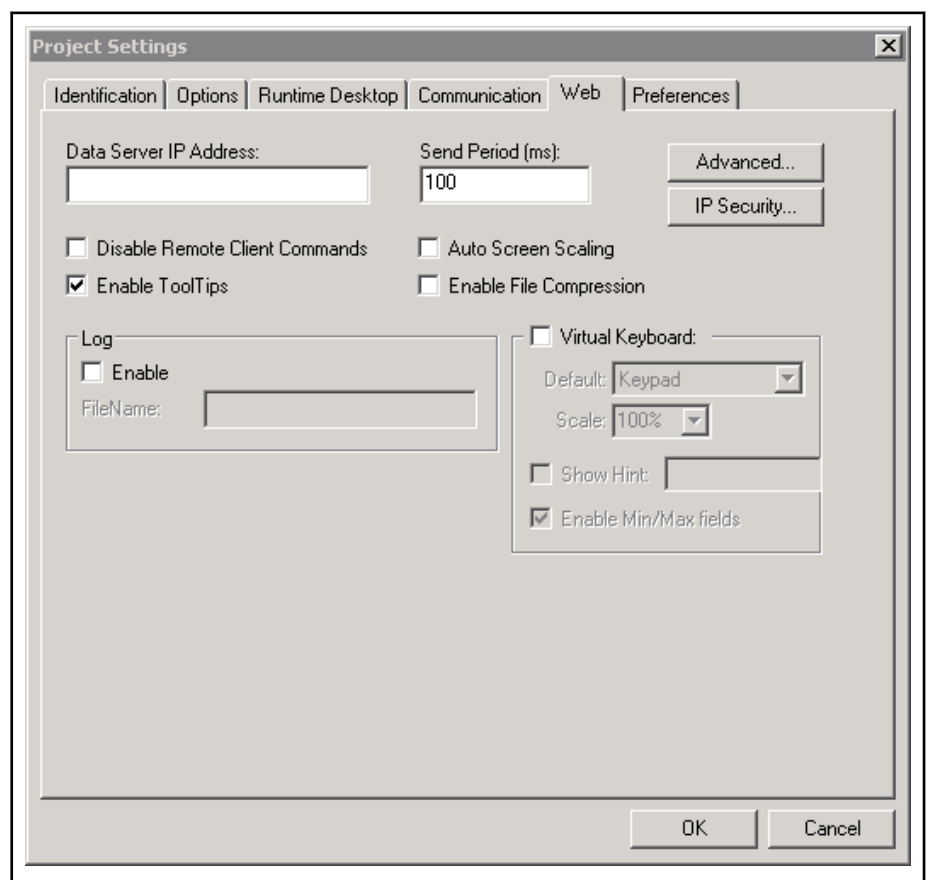


Fig. 4-62: Dialog: Project settings (tab: Web)

Use this tab to specify the parameters for the Thin Client:

- **Field: Data server IP address**

Entering the IP address (or the host name) of your data server station. The data server station is the computer or the device on which the TCP/IP server module of WinStudio is running.

- **Field: Sending period in ms**

Enter a value to specify the update rate (in milliseconds) to exchange data between the server and the Web Thin Client stations.

Overview

The default value is 1000 (milliseconds). A lower value can be entered in this field to increase the update rate between server and Web Thin Clients. If a lower value is entered, it can result in increased traffic in the network (more accesses to network) if the tags change continuously (quicker than 1 second).

- Check box: **Blocks remote client commands**

Enabling this box prevents that a remote Client passes commands from the Web Thin Client to the server. If this option is enabled, the Web Thin Client can read data from the server but cannot send any data (tag values, set points) to the data server. The Web Thin Client station is rendered to a "read only station" in this case.

- Check box: **Allow ToolTips**

By enabling this box, ToolTips that have been configured for objects on the screens, can be displayed if they are to be viewed on the Web Thin Client (web browser).

- Check box: **Auto screen scaling**

Enable this box to auto size the screens in a web browser window. When using this option, the web browser size adjusts to the screens irrespective of the screen resolution.



The option automatic screen scaling does not apply to web browsers running in Windows CE operating systems.

- Check box: **Allow file compression**

Enable this box to compress files saved in the Web folder of the application. This option can be used to decrease the download time, especially if you only have a slow connection between server and Web Thin Client.

Project status

The runtime components have to be enabled to correctly execute some WinStudio functions. An overview of all WinStudio runtime components can be called via **WinStudio ► Status...**

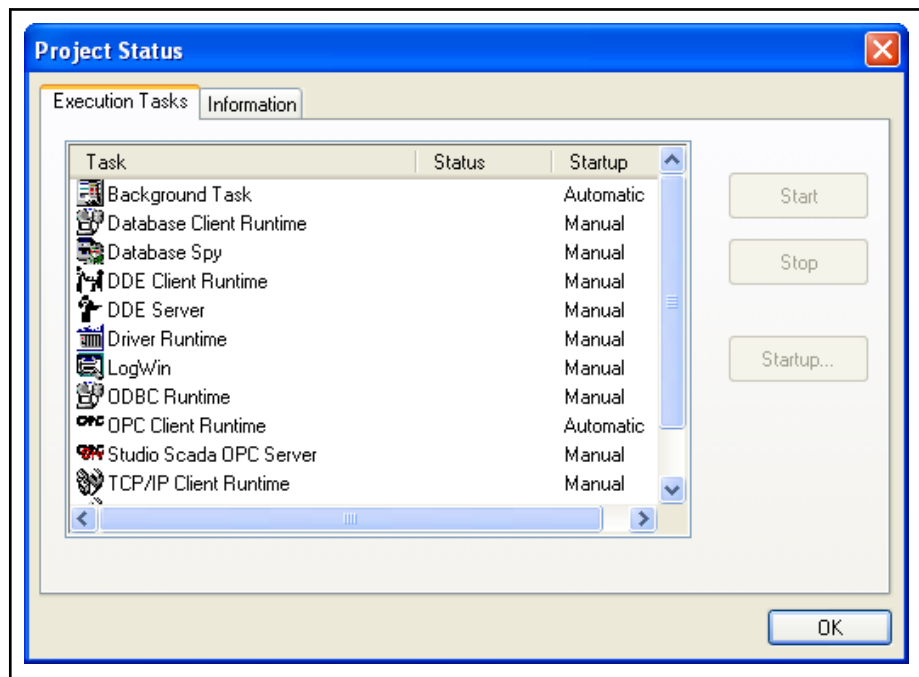


Fig. 4-63: Dialog: Project settings (tab: Execution tasks)

All available WinStudio runtime components are displayed in the status dialog. The status as well as the start mode are displayed for each runtime component. By default, the following runtime components are enabled upon start of IndraWorks Operation:

- Background task
- OPC Client runtime
- TCP/IP server

Load WinStudio project

The current WinStudio project including all WinStudio objects can be replaced by another WinStudio project to use an existing WinStudio project as template. To replace the current WinStudio project, execute the function **WinStudio ► Load Project...** . A file browser opens with which a project file (file extension "*.app") can be selected. After the file selection has been confirmed, all current WinStudio elements are deleted and new elements of the selected project are inserted.

WinStudio Screens

Creating and Editing WinStudio Screens

To create a WinStudio screen, please select **WinStudio ► New View**. A new element with the name "Screen" is added to the "WinStudio" folder. This name can be changed now or later.

To edit an available WinStudio screen, double-click on the element in the project tree to open the WinStudio screen editor.



Some elements can only be used if the WinStudio-Engineering license 1,5k or higher is activated. Elements with a license are disabled in case of an insufficient license.

When opening a WinStudio screen, additional symbol bars with frequently used function are displayed in the default setting. The default setting can be changed via **Tools ► Adjust...** and the symbol bars can be hidden or more symbol bars can be added. All editing functions of a WinStudio screen are also available via the menu line of IndraWorks Engineering via **WinStudio screen** .

Screen Properties

More options to change the screen size and to use a background graphic are available in the screen properties. The screen property dialog can be opened via the context menu of the screen editor via the menu item **Screen properties** .

Overview

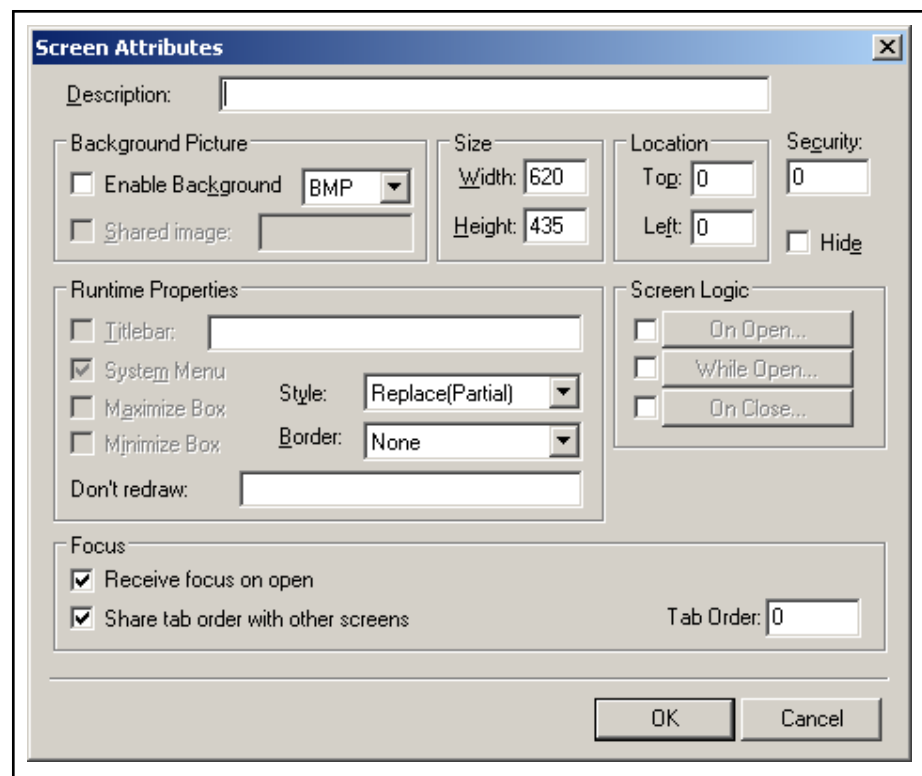


Fig.4-64: Attributes of a WinStudio screen

Dynamic object properties

Some properties (color, size, position) of WinStudio objects that can be inserted in a WinStudio screen can be changed dynamically. Moreover, functions can be configured that can be executed when enabling WinStudio objects.

Several dynamic properties can be assigned to a WinStudio object by enabling them individually. The following dynamic properties are available:

- **Command**
Function that is to be executed while enabling the element
- **Hyperlink**
opens when enabling the hyperlink
- **Bar diagram**
represents a variable value as bar diagram
- **Text input/output**
shows the value of a WS variable or applies the entered value
- **Color change**
changes the color depending on a WS variable
- **Position**
changes the X and Y position of an object depending on a WS variable
- **Size**
changes width/height of the object depending on a WS variable
- **Line rotation**
rotates a line depending on a WinStudio variable

Object Properties and Variable Selection

The properties of a WinStudio object can be edited by means of the property dialog. Basic properties (e.g. line width) can be edited. The scope of the offered properties depends on the respective object type.

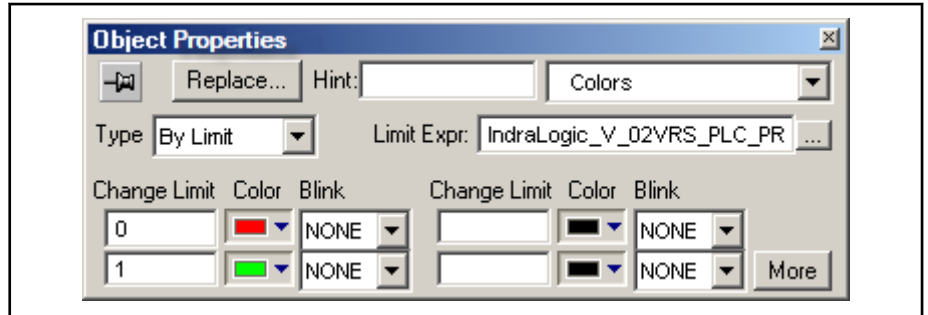


Fig.4-65: Object properties

Variables of the following areas can be selected in the selection dialog:

- Application
- Divided
- System

WinStudio Variables

WinStudio offers the possibility to execute action (e.g. calculations) depending on a variable. Results of calculations can be assigned to variables and WinStudio variables can be linked to process variables.

In the "Variable" folder, the following variable categories are available:

- **Internal variables**
Predefined, unchangeable variables
- **Distributed variables**
Variables linked to a process variables
- **Application Variables**
individual, user-specific variables

Classes WinStudio provided the following data types to individual variables:

- integer
- Bool
- real
- String

Moreover, individual data types, so-called "classes" can be defined that are composed of standard types.

Overview

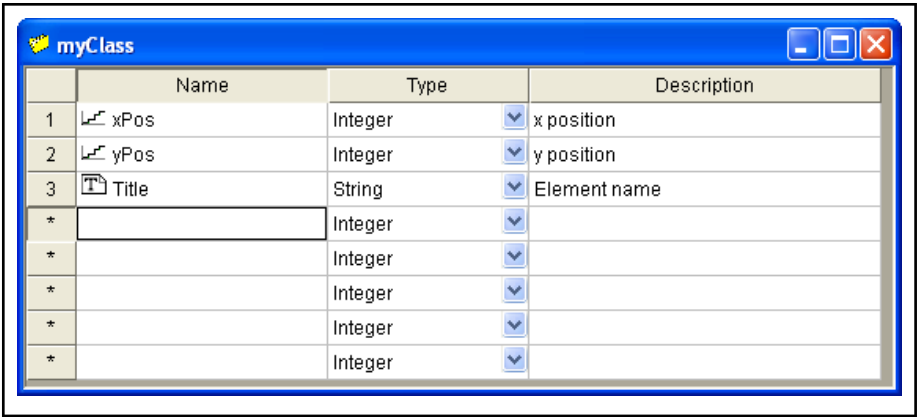


Fig.4-66: WinStudio class editor

In the class editor, individual element names can be added to the "Name" column and a data type can be assigned under "Type". After the new data class has been saved, they can be used for WinStudio application variables.

Communication

WinStudio variables can be linked to PLC variables to show the value of the PLC variable in a WinStudio screen, to use it for calculations or to change it. WinStudio variables are linked to PLC variables via so-called driver sheets. A driver sheet contains different settings, apart from the variable list, that are required for the communication with the PLC. The driver sheet designation consists of a unique number and an optional description text. OPC as well as BRWS driver are available for the communication.

Linked Symbols

WinStudio Library

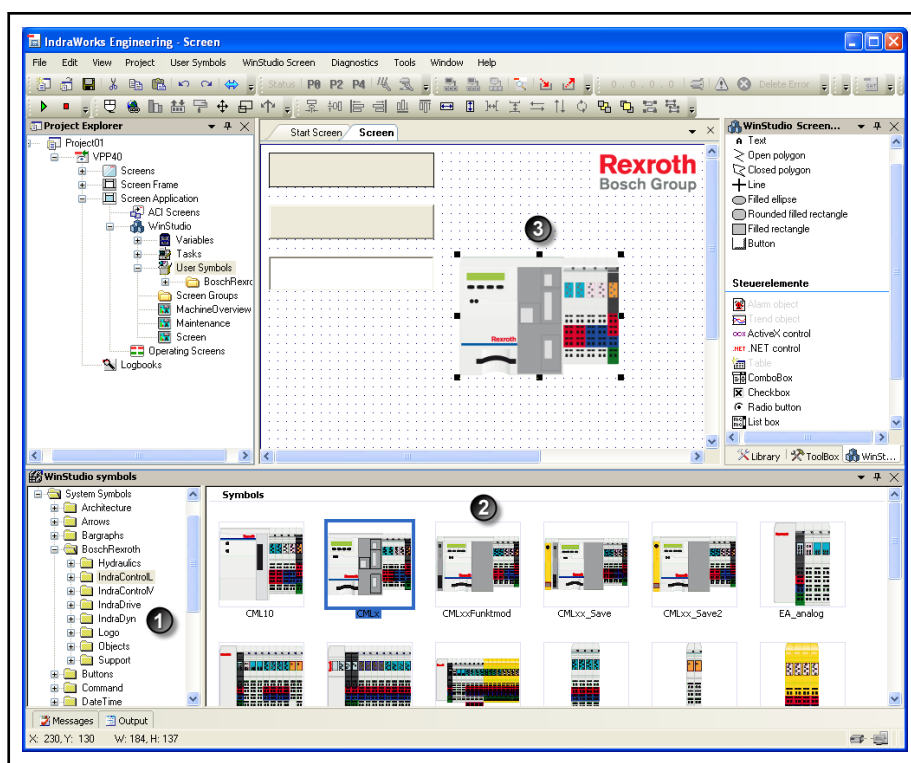
WinStudio provides a comprehensive library of symbols that can be inserted in WinStudio screens.

The WinStudio library is represented in a tool window of IndraWorks Engineering (see next figure). The library consists of many folders in which the WinStudio symbols are contained.

Insert symbol

To insert a symbol in a WinStudio screen, open the WinStudio screen. Subsequently, select a symbol by selecting it in the preview window. By clicking on the WinStudio screen editor, the symbol can be inserted in the desired position.

Overview



- 1 Folder with libraries
- 2 Available symbols of a folder
- 3 WinStudio screen editor

Fig. 4-67: WinStudio symbol library

Parameterize symbol

Each WinStudio symbol has several parameters that have to be adjusted correspondingly. The parameterization dialog opens by double-clicking on the symbol in the WinStudio screen. All symbol parameters are sorted by category and are displayed by clicking on the category. Individual parameters can be changed by clicking on the "Value" column and entering a value. A value or a WinStudio variable can be entered in this column.

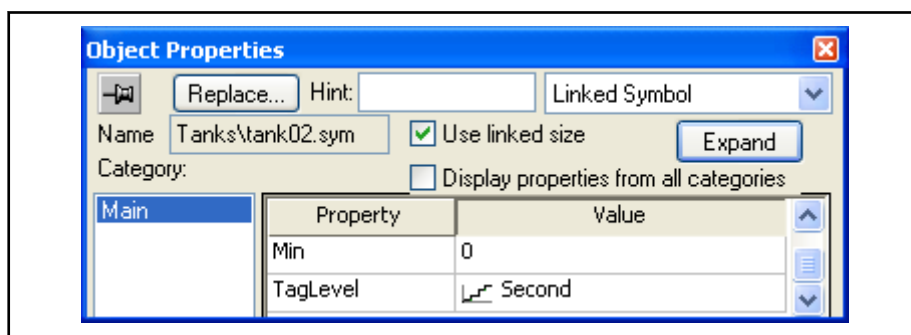


Fig. 4-68: Parameterization dialog of a WinStudio symbol

User Symbols

Customized symbols can be created in addition to pre-defined WinStudio symbols. If an object is to be used for different purposes in a WinStudio project, creating a new symbol can be really helpful. If a symbol is changed, this is applied to all WinStudio screens in which the symbol is used. The symbol does not have to be inserted in the WinStudio screens again.

Creating a symbol

To create a symbol, insert one or several objects in a WinStudio screen, highlight them and click on **Group** in the context menu. Subsequently,

Overview

execute **Generate linked symbol** in the context menu and enter a symbol name in the file dialog.

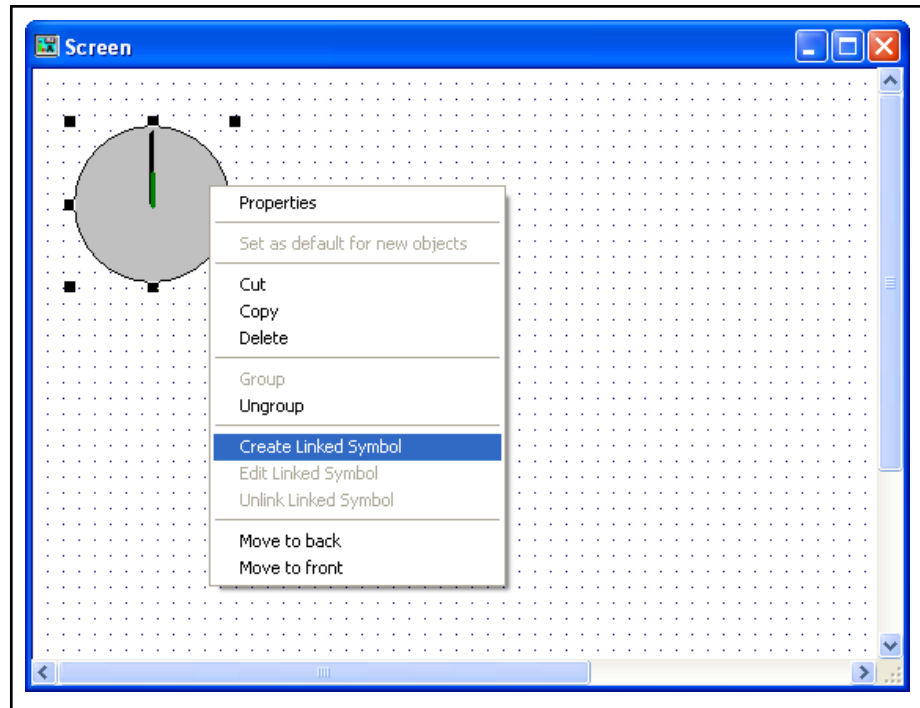


Fig.4-69: Creating a user symbol

Apply symbol to library

A created WinStudio symbol is initially only available in the same project. If it should also be used in other projects, apply it to the WinStudio library by clicking on the element in the project tree under **WinStudio ► User symbols** and by subsequently executing the function **Apply to WinStudio Library** in the context menu. The symbol is copied to the "User Symbols" folder in the WinStudio library.

Password-protect symbol

You can assign a password to customized user symbols so that only authorized users can edit the symbols. A protected user symbol can thus be used and parameterized in WinStudio screens but the internal functionality cannot be edited.

Upon opening a protected user symbol, an input dialog is displayed, prompting the user for a password. The editor only opens when the correct password is entered.

WinStudio Tasks

Mathematic Sheets

A WinStudio mathematic sheet is used to program sequences, variable assignments and the execution of other WinStudio functions.

Math Sheets:1

Description:
Tag Counter

Execution:
Execute

	Tag Name	Expression
1	Counter	Counter + 1
*		
*		
*		
*		
*		

Fig.4-70: WinStudio mathematic sheet

The mathematic sheet is executed cyclically if the variable or the expression in the "Execution" field is TRUE.

Alarms

By means of alarms, messages can be defined that are output if a variable exceeds or falls below a value.

1: Fill level observation

Description:
Fill level observation

Group Name:
Email Settings... Advanced...

On Line

- ☒ Display in Alarm Controls
- ☒ Ack Required
- ☐ Beep
- ☐ Send to printer

History

- ☐ Save To Disk
- ☐ Generate Ack Messages
- ☐ Generate Norm Messages

Colors in Alarm Controls

☒ Enable

Activation

FG ■ BG ■

Acknowledgment

FG ■ BG ■

Normalization

FG ■ BG ■

	Tag Name	Type	Limit	Message	Priority	Selection
1	fill_level	Hi	50.0000...	Container full	0	
*		HiHi				
*		HiHi				
*		HiHi				
*		HiHi				
*		HiHi				

Fig.4-71: WinStudio alarm editor

Alarm messages can be displayed by means of the alarm control element in WinStudio screens.

Recipes

In WinStudio, variable values can be saved in files. Variable values can also be loaded from files by means of so-called recipes. A WinStudio recipe is a list of variables and the assignment of variables.

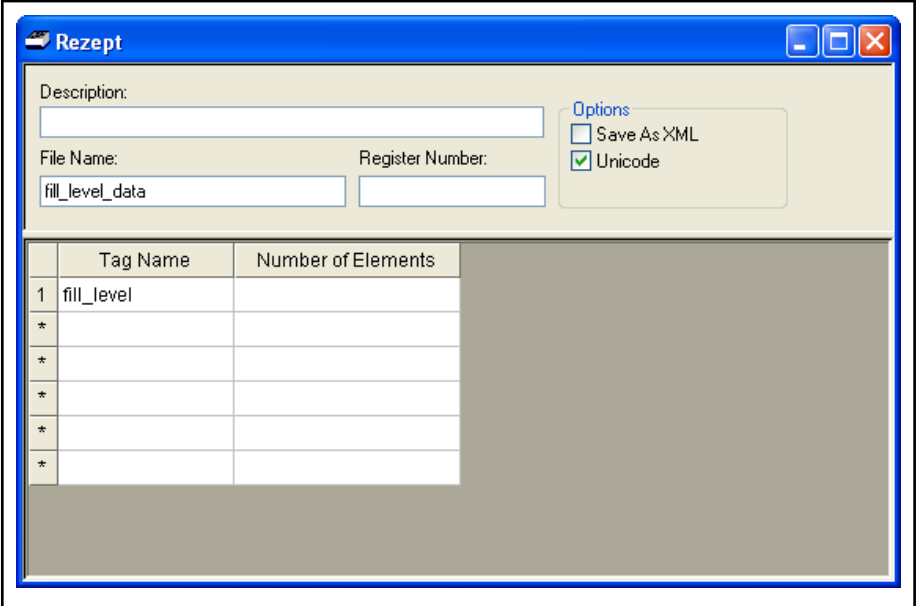


Fig.4-72: WinStudio recipe editor

In the "Tag Name" column of the recipe editor, the variable names can be entered whose values are to be saved or loaded. If the variable is an array, the element number has to be entered in the "Number of elements" column.

Loading and saving recipes

To save variables in a recipe, insert the following cal (e.g. as interface command or in a mathematics sheet): `Recipe("Save:<RecipeName>")`. If the function is executed, the current variable values listed in the recipe are written to a file. Values that have been saved in a recipe can also be loaded again by using the `Recipe("Load:<RecipeName>")` command. In this case, the current variable values are overwritten with the recipe values.

Reports

A WinStudio report is used to create a documentation that can contain text as well as current WinStudio variable values. The report can be saved in a file or printed out.

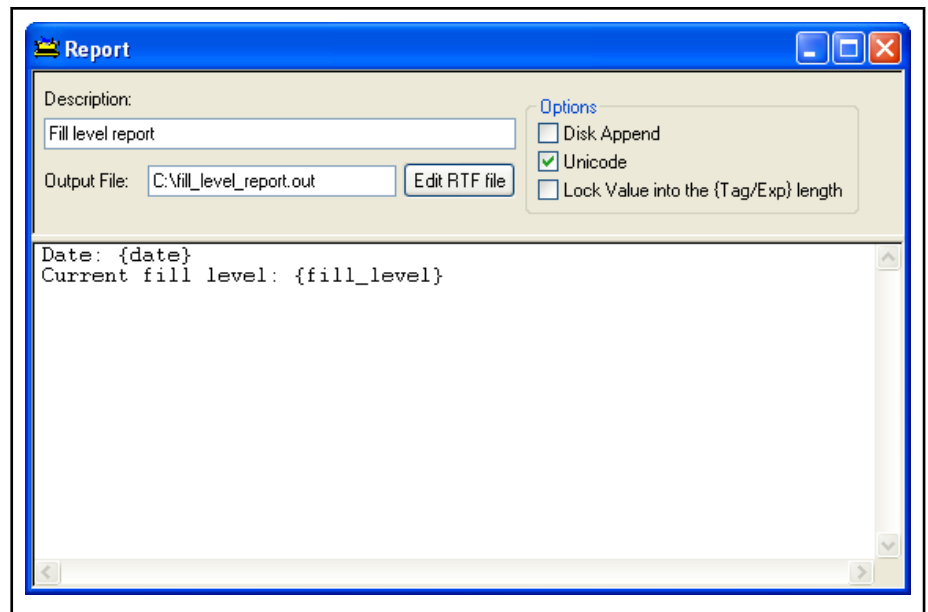


Fig. 4-73: WinStudio report editor

In the field "Output File", the report file name can be specified. In the lower editor area, text can be entered that is to be written in the report. Variable values can also be output with the help of the following syntax: {<Variable-name>}.

Output report

A report is generated during runtime by IndraWorks Operation by executing the Report("<Reporttype>:<Reportname>") command.

Parameters of the report command:

- **Report type**
"Disc" to output it in a file or "Prn" for printing the report on a standard printer
- **Report name**
Name of report in "Reports" folder

Trends

WinStudio trend data serve to record WinStudio variables. These variables can be represented in a WinStudio screen by means of a trend object to monitor the timing sequence of variables.

Details about the configuration of trend data objects and to visualize trend data can be found in the WinStudio online help.

Sequences

Actions can be executed cyclically by means of WinStudio sequences. For example, it is possible to automatically generate a report each day or to reinitialize a variable every 30 seconds.

Overview

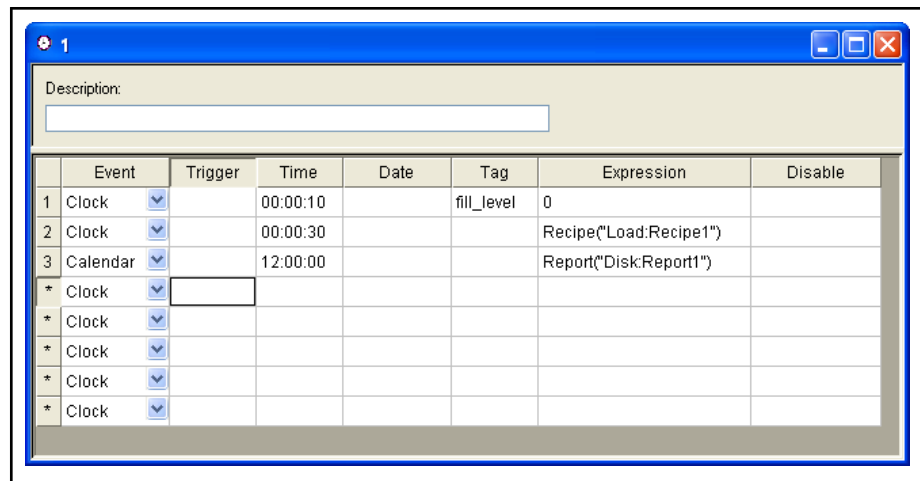


Fig. 4-74: WinStudio sequence editor

One event and one action can be defined in the sequence editor per line. In the "Event" column, the interval type of each event can be specified.

The following options are available:

- **Clock/time**
The event occurs in an interval of one second up to 24 hours.
- **Calendar**
The event occurs in an interval of >24 hours.
- **Change**
The event occurs when changing the variables in the "Trigger" field.

The interval variable can be specified by means of the "Time" and "Date" column:

- **Time**
Interval variable in hh:mm:ss for "Clock/time" event; time at event "Date"
- **Date**
Event date. If the field is blank, the event occurs on a daily basis

Each event can be assigned an action by means of the "Variable" and "Expression" columns. A WinStudio function or a value can be entered in the "Expression" column that is to be assigned to a variable.

ODBC

By means of ODBC driver sheets (Open Database Connectivity), WinStudio variables can be linked to external databases. Alarms, trends and events support the connection to an external database by means of ODBC.

A detailed description of the ODBC configuration can be found in the WinStudio online help.

Scripts

Overview

WinStudio supports the creation and execution of user-defined functions in VisualBasic scripts. The VisualBasic script language is a simple and flexible tool to program functions and sequences. An introduction to VB (VisualBasic) and a language reference can be found in the MSDN (Microsoft Developers Network).

Global Procedures

Some VB function can be declared as "Global Procedures" and a function library can be created. Functions that are declared under "Global Procedures" can be called in all WinStudio objects with VB support (e.g. as command on interfaces).

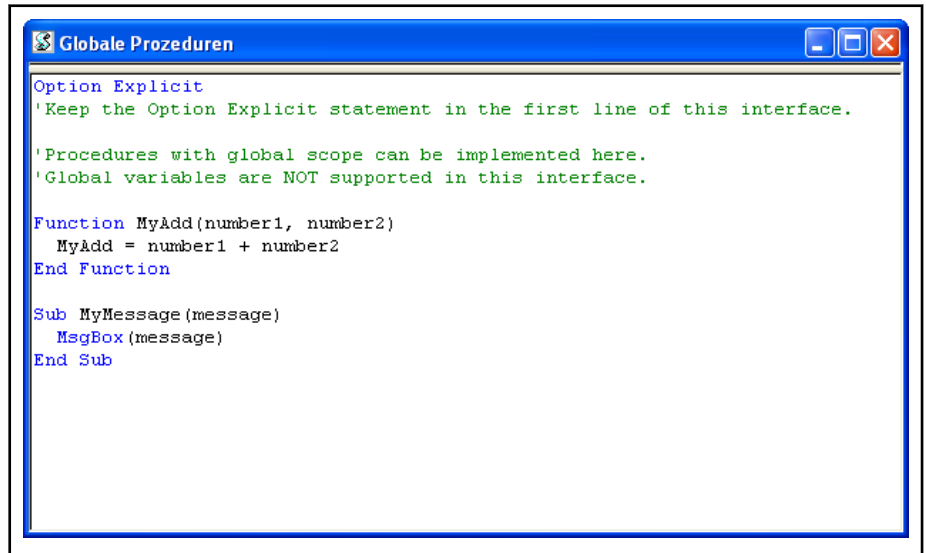


Fig. 4-75: Dialog: Global Procedures (editor)

A user-defined number of functions and subfunctions can be declared in the editor for global procedures.

Start Script

In the "Start script" element, VB function and user-defined global procedures can be called that are to be executed once upon the start of the application. The start script can particularly be used to initialize variables.

Graphic Script

The VB script "Graphic script" can be used for one-time execution of VB statements upon starting or closing the WinStudio graphics module or for continuous execution during the runtime of the graphic module. The graphic script already contains all required subfunctions that can be inserted in the user-defined VB code. To correctly execute functions in the graphic script, the function names cannot be changed. More function can be added that can be called in the pre-defined graphic functions.

Overview

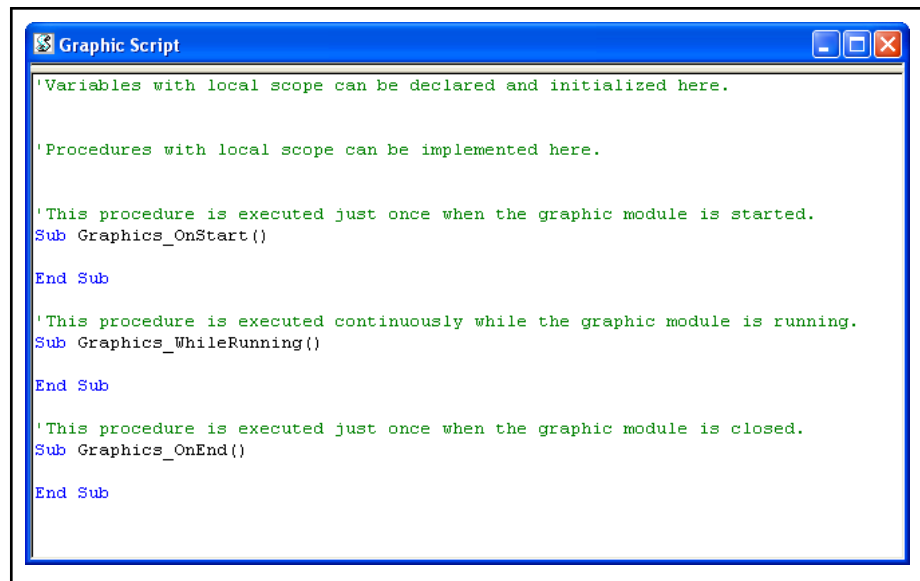


Fig.4-76: Dialog: Graphic script (editor)

Screen Scripts

VB statements can be programmed for each WinStudio screen. The statements can be continuously executed upon the opening or closing of the screen while the WinStudio screen is still opened. The screen scripts already contain all required subfunctions that can be inserted in the user-defined VB code. The names of the pre-defined functions cannot be changed.

A VB script exists for each WinStudio screen that can be opened and edited in the context menu of the opened screen editor via **Screen Script**.

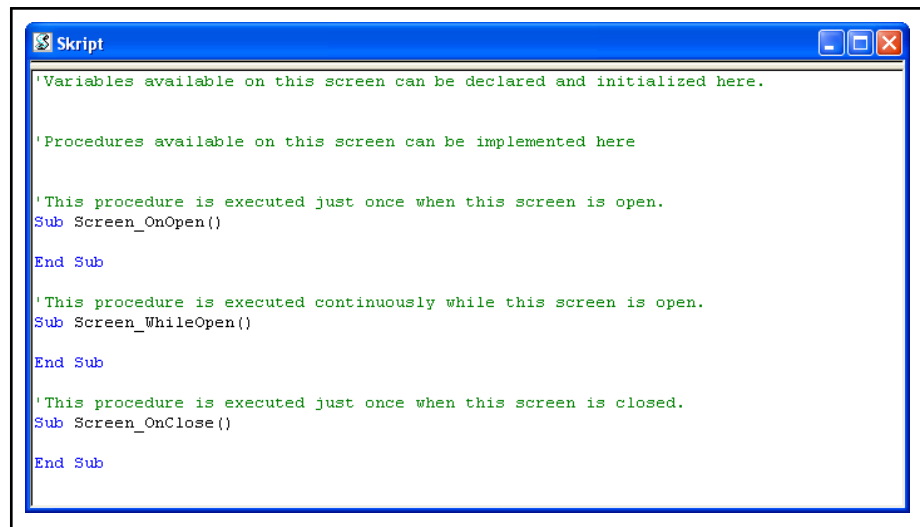


Fig.4-77: Dialog: Script (screen script editor)

User Scripts

In addition to the pre-defined VB scripts ("start scripts", "graphic script" and "screen script"), any number of VB scripts can be added in which VB scripts can be inserted. Functions cannot be declared in user scripts.

User scripts contain a continuous number that specifies the execution sequence of the user scripts. To exclude a script from the execution during runtime, enter a WinStudio variable or expression in the "Execution" field. In this

case, the script is only executed if the variable or the expression has a value greater than zero.

WinStudio Tools

Verify Application

Variables can be used in all WinStudio elements that are required for the respective function. During the configuration, inconsistencies can occur by deleting or renaming variables so that the affected WinStudio functions and element do not function correctly anymore.

By means of the function "Verify Application", it can be checked if variable names are used in WinStudio elements (mathematics sheets, WinStudio screens, scripts, etc.) that are not contained in the variable database.

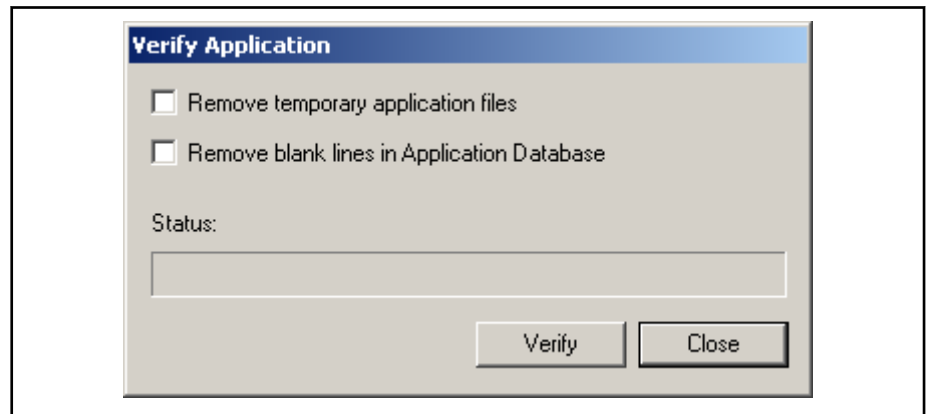


Fig. 4-78: Verify application

Warnings and error messages are output in the output window of IndraWorks Engineering after the verification dialog is closed.

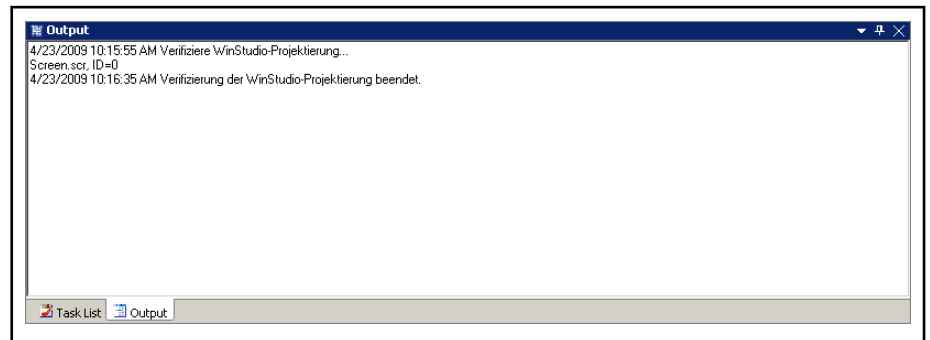
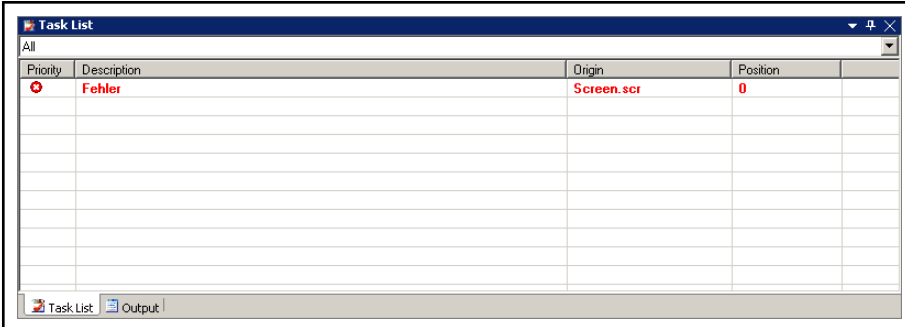


Fig. 4-79: Output of messages of the function "Verify application"

If errors occur during the verification, they are output in the task list of IndraWorks Engineering. The error source is entered in the "Origin" column. The object ID number is displayed in the "Position" column. By double-clicking on the error line, the editor opens and the incorrect element is automatically selected. That way, all inconsistencies can successively be corrected and the function "Verify Application" can be restarted if required.

Overview



The Task List window displays a table with the following data:

Priority	Description	Origin	Position
1	Fehler	Screen.scr	0

Fig.4-80: Error list of the "Verify Application" function

Start Application

The function of individual WinStudio screens can be checked using the function "Start application" before enabling the entire package in IndraWorks Operation. The test window can be opened via **WinStudio ► Start application**.

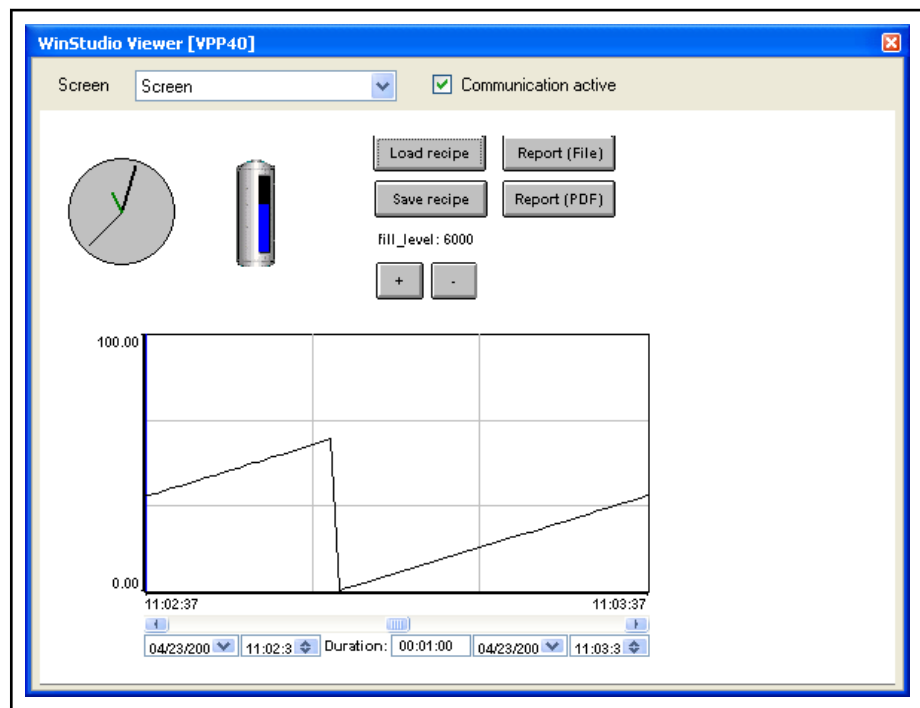


Fig.4-81: Testing WinStudio screens

All screens are contained in the current WinStudio application in the "Screen" selection list. When selecting a screen, it is displayed in runtime mode and the variable display, commands on interfaces or the execution of scripts can be tested.

Database Spy

By means of the WinStudio database spy, the value and the status of individual WinStudio variables of an active WinStudio application can be monitored. In the input field of the database spy, the WinStudio variable name can be entered and applied to the list of variables to be monitored by clicking on **Add**. The current value as well as the connection status is displayed for each variable in the variable list.

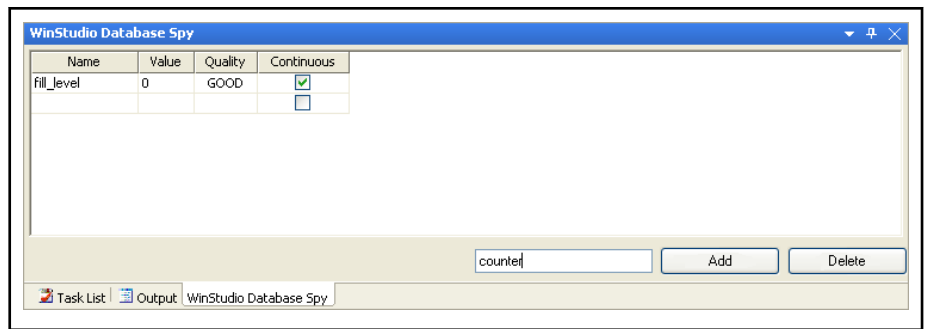


Fig. 4-82: WinStudio database spy

LogWin

WinStudio LogWinTo can be used to analyze WinStudio LogWin application sequences and to search for errors. In LogWin, OPC communication messages, individual trace messages and the recipe execution can be protocolled.

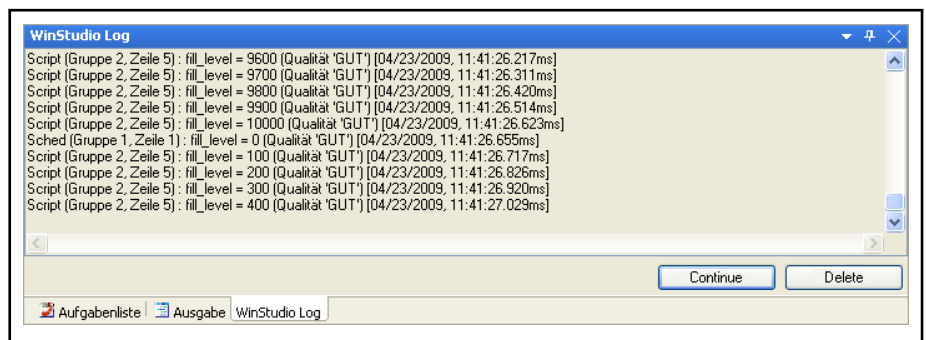


Fig. 4-83: WinStudio message output in WinStudio LogWin

In the context menu of the LogWin window, a configuration dialog can be opened via **Settings** with which the messages to be recorded can be opened. Furthermore, WinStudio variables can be selected for which a message with the current value is to be output when variables are changed.

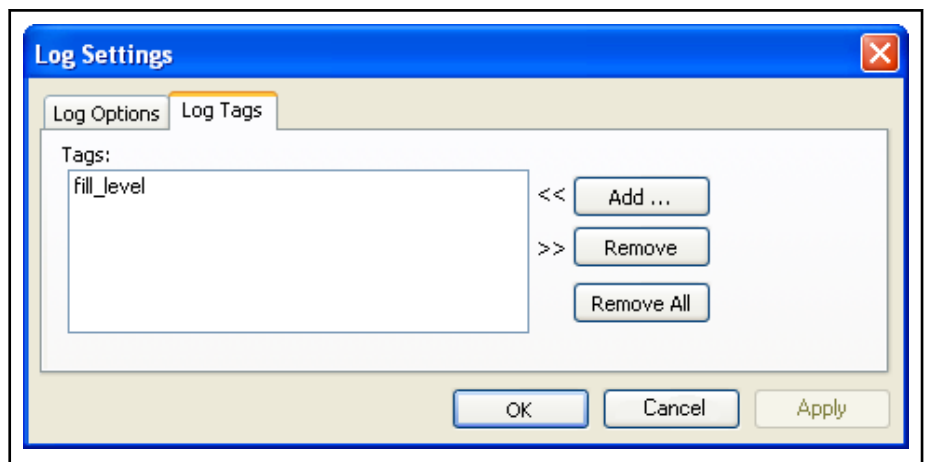
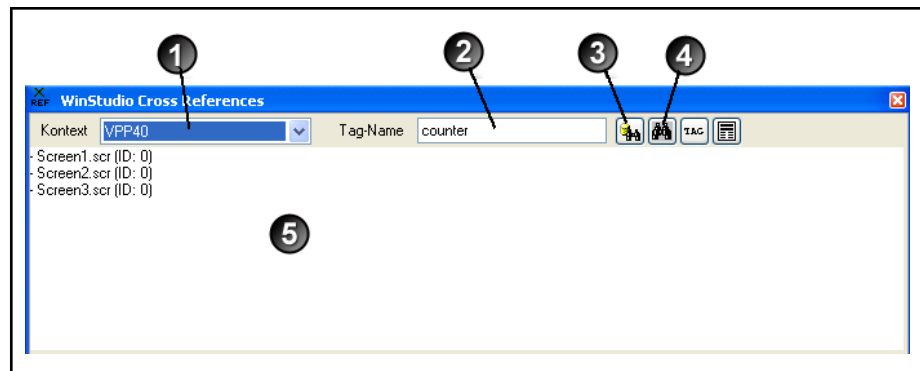


Fig. 4-84: Monitoring WinStudio variables in LogWin

Cross Reference Search

Use the cross reference search to find certain WinStudio variables.

Overview



- 1
- 2 Variable
- 3 Search in file
- 4 Search in project
- 5 Display

Fig.4-85: Cross reference search of WinStudio variables

Variable Import Wizard

Often, it is practical to apply variable lists from another WinStudio application or a database to avoid that they have to be configured again manually. Applying variables is supported by the import wizard.

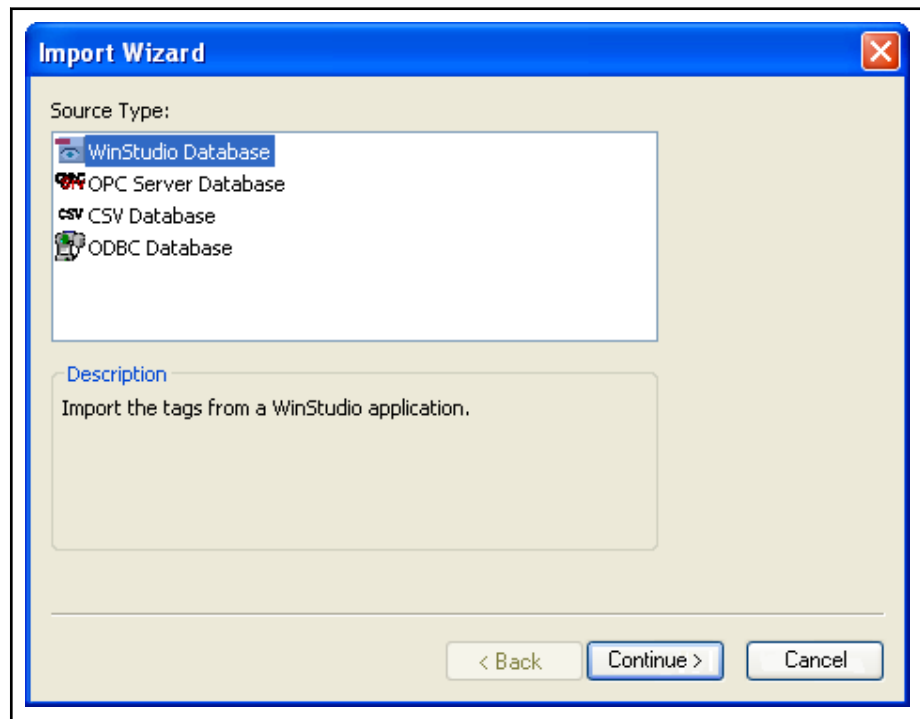


Fig.4-86: Selecting a data source in the WinStudio import wizard

Export/Import

The "WinStudio" folder as well as all corresponding subelements support the Export/Import functionality in IndraWorks. By means of this functionality, the entire WinStudio project or only parts of can be exported and imported in another IndraWorks project.

Please note the following characteristics when using the Export/Import function on WinStudio elements:

- The WinStudio variables are determined and automatically exported for each WinStudio element. When importing this WinStudio element, the required WinStudio variables are inserted again. If the variable to be imported already exists in the new project, all corresponding variable properties (e.g. description text, initial values) are overwritten. If the variable type differs, the variable remains unchanged. If you still want to import the variables, change the variable type or rename the variable and repeat the import.

If an existing variable has been overwritten during the import or if the new variable could not be imported, the import dialog displays a corresponding message.

- If the WinStudio variable to be exported is a user-defined WinStudio data class, the corresponding class definition also has to be exported so that the variable can be created correctly. Therefore, a query is displayed during the export, asking the user if he wants to import the required class if the class has not been selected yet. It is recommended to confirm the query with "Yes".

WinStudio Licensing

Introduction

The graded license model permits an optimized adjustment to the required number of variables and the required functional scope. The larger license package covers the functional scope of the smaller package. An upgrade to the next higher license class is possible.

The WinStudio licensing procedure has been changed with the introduction of WinStudio 7.x (from IndraWorks 09VRS). In the following, please find a description of the new license management.

WinStudio License Overview

The following licenses are available:

- **WinStudio Lite**
This license is part of the IndraWorks installation. The user can create and execute WinStudio projects with a maximum of 500 variables (tags). Moreover, all CE devices are equipped with a Lite license to execute Lite runtime projects. With this license, alarms, recipes, reports, trends as well as TCP/IP and DDE driver sheets are not executed and cannot be created or edited.
- **WinStudio Runtime 1K5 / 4K / 64K / 512K (RUN licenses)**
For WinStudio runtime projects, licenses with a maximum of 1.500, 4.000, 64.000 and 512.000 WinStudio variables (tags) are available
- **WinStudio Engineering and Runtime 1K5 / 4K / 64K / 512K (RUD licenses)**
By means of this license, WinStudio projects with a maximum of 1.500, 4.000, 64.000 or 512.000 WinStudio variables (tags) can be created and executed
- **WinStudio demo license**
With this license, the functional scope of WinStudio can be tested within a period of 30 days. After the test period expired, the last valid license (e.g. Lite license) is active
The demo license can only be enabled once.

Overview



Only 1K5 and 4K runtime licenses are available for the activation on an IndraControl VEx device.

Activating a WinStudio License

The individual "WinStudio Runtime" and "WinStudio Engineering and Runtime" licenses are subject to additional costs and have to be activated by entering a license key. The license sheet contains your personal license key.

To activate a license in IndraWorks Engineering, go to **Tools ► Options ► General Information ► Software Licenses** in the licensing dialog and select "WinStudio". The activation dialog can be opened by clicking on **Licensing...** and the activation key can be entered.

After the license has been activated, it may be required to restart WinStudio Engineering or IndraWorks Operation.

Saving the license on a dongle

The activation key is only valid for one PC. To be able to use a license on several PCs, the license can be saved on a dongle. The license is valid as long as the dongle is plugged in a USB interface of the computer. Windows CE licenses (VEx) cannot be transmitted to a dongle.

License activation on VEx devices

To enable a WinStudio license on a VEx device with a Windows CE operating system, open the licensing dialog on the VEx device in the Windows start menu via **All Programs ► Rexroth ► WinStudio ► License management** and enter the activation key. The licensing software is already installed on new VEx devices. If you already have a VEx device without pre-installed licensing software, install the IndraWorks visualization software on the VEx device and enable the WinStudio license.

To save the license setting, start IndraWorks Operation on a VEx device and subsequently back up the Windows registry.



The license activation via the WinStudio remote agent is not possible anymore with the new licensing procedure. Use the IndraWorks licensing dialog to activate the license (as described above).



The transmission of the license key via the integrated WinStudio function is not possible anymore with the new licensing procedure. The license key has to be directly entered at the VEx device.



The IndraWorks license component for Windows CE is already installed on new devices. If you want to update an already existing VEx device with the new software, install the runtime component on the VEx device via the context menu of the VEx device in IndraWorks Engineering with the function **Install Software**. The link might not be given. In this case, start the licensing dialog via `\StorageCard\Rexroth\WinStudio\bin\IndraWorks.LicenseImport.CE.exe`. The licensing component requires .Net CompactFramework 2.0 so that only the firmware has to be updated.

Consequences of a Missing License or a License of a Too Low Class

When opening a project with the WinStudio editor that has been created on another computer with a higher class WinStudio license, the elements in Win-

Studio Engineering requiring a license are not activated. This means that these elements can neither be created nor edited in this case.

When starting IndraWorks Operation, a message is displayed informing the user that the available WinStudio license is insufficient for the current project. No WinStudio screens are displayed in case of an insufficient or missing license.



If you still want to use the current project, activate the demo license. Use the IndraWorks licensing dialog to activate the license (as described above).

Multilingualism

WinStudio user screens can have multiple languages. All texts in the WinStudio screen are displayed in the selected project language during the IndraWorks Operation runtime.

It is required to create a text file for each language.



The text file extension is ".tra". From WinStudio version 6.3, a new file format will be available (.csv). When saving data, the file type "All Files (*.*)" has to be selected instead of "Text Files (*.csv)".

WinStudio Applications

Overview

Optionally, applications can be created in IndraWorks that are exclusively based on the WinStudio runtime environment. This is required if you want to use the WinStudio web functionality. Creating and enabling WinStudio applications differs from the previously described application within the IndraWorks runtime environment. These differences are explained in the following.

Creating a WinStudio Application

To create a WinStudio application, insert an VPP or VEx device from the device library to an IndraWorks project. Select the "WinStudio application" type in the HMI wizard. After confirming the entries by clicking on **Complete**, the corresponding WinStudio elements are created in the project tree. Elements such as F-, M- and OP-panels, operating screens, etc. requiring an IndraWorks runtime environment are not offered for selection.

User Management

The WinStudio user management can be used to protect different WinStudio functions in WinStudio applications. The WinStudio user management is configured via the function **WinStudio ► Tools ► Safety settings**. The safety system can be enabled in the configuration dialog. Groups and users can be created and the corresponding rights can be assigned. For more information on how to use the WinStudio user management, please refer to the WinStudio online help "Studio Technical Reference".



Users and groups that have been configured in the IndraWorks user management are not available in the WinStudio user management. If required, they have been created again in the WinStudio user management.

Overview

Web Functionality

WinStudio applications can be visualized by means of a web browser at run-time. Applications have to be saved in HTML format.

To save a WinStudio screen in HTML format, open the context menu of the WinStudio screen and execute the function "Save as HTML". Optionally, the function "Save all documents as HTML" can be used to save all existing WinStudio screens in HTML format.

After the configuration and saving of all required WinStudio screens in HTML format, the web server has to be configured. Open the WinStudio project settings and select the "Web" tab. Information about configuration and use of the WinStudio web server can be found in the WinStudio online help "Studio Technical Reference".



The functions "Save as HTML" and "Save all documents as HTML" as well as the "Web" tab in the WinStudio project settings are only available for a "1,5 k" WinStudio license or higher.

Transmitting the Application and Activation on the Target Device

To transmit the WinStudio application to a target device, start the WinStudio remote agent. Subsequently, **WinStudio target platform** "Execution platform" can be opened via this WinStudio context function. In this dialog, it can be selected which target station you want to connect to. In the "Local" setting, no other inputs are required. The WinStudio runtime environment can be started or exited by clicking on **Start** and **Stop** on the "Application" tab.

If you select the option "Network IP" on the "Target" tab and enter a valid IP address, you can connect to the target device by means of the remote agent by clicking on **Connect**. If the connection was established successfully, the suggested target directory can be changed by means of the directory browser on the "Application" tab. The **Send to target** interface copies all files of the current WinStudio application to the selected target device directory. The **Start** and **Stop** buttons start or stop the WinStudio application on the target device.

Further Information

WinStudio offers comprehensive functions to create user screens. The complete functional scope is described in the "saw.IW-HMI.configuration_WinStudio_V12.xml" and the "WinStudio Online Help".

4.4 Communication Interfaces

4.4.1 Overview

The following table offers an overview of the combination possibilities of visualization devices, visualizations and communication protocols.

Device and visualization	Communication protocol					
	OPC.DA	OPC.UA	OPC.lwS CP	BRWS	Mod-busTCP	IndraLogic gateway
VCP/VCH						
VI Composer				X		
VEP/VEH (WinXPe)						

Device and visualization	Communication protocol					
	OPC.DA	OPC.UA	OPC.lwSCP	BRWS	Mod-busTCP	IndraLogic gateway
WinStudio	X	X	X*	X		
External visu	X*	X	X*	X	X	
VEP/VEH (WinCE)						
WinStudio	X	X		X		
External visu	X	X		X	X	
VSx/VPx						
WinStudio	X*	X	X*	X		
VI-Logic						X
External visu	X*	X	X*	X	X	

Fig.4-87: Overview of the combination possibilities of visualizations

* Available after the installation variant "Communication - IndraLogic, IndraMotion MLC" has been installed using the IndraWorks installer.

4.4.2 OPC.lwSCP

Properties

OPC.lwSCP is the Bosch Rexroth individual standard OPC server.

It is used for communication connection of IndraWorks projects in connection with the standard operating interface "IndraWorks Operation" on pc-based hardware systems (see [chapter4.2 Visualization Devices, page 27](#)).

It can also be used for WinStudio and "Third Party" applications. Requirement: Use of the SCP communication platform that uses IndraWorks to communicate with the control systems.

If OPC.lwSCP is used without IndraWorks Operation Desktop, the OPC.lwSCP has to be configured for devices that it has to connect to. The following chapter [SCP-OPC Configurator, page 85](#) does only apply to this use case.



Detailed information about the OPC communication, the items that can be used and a possibly required Client programming are available in the DOK-IM*ML*-OPC*COM*Vxx-APxx-EN-P documentation

SCP-OPC Configurator

General

The SCP-OPC Configurator is used for declaration of the devices which are provided by SCP via OPC. The SCP-OPC Configurator allows the creation of configuration files and their designation as active configuration. When starting the OPC server, the active configuration is evaluated and the relevant SCP devices are created. Selected devices are highlighted in blue. Modifications to an active configuration only take effect after the next restart of the OPC server.

Overview

The SCP-OPC Configurator displays the list of devices contained in an SCP-OPC configuration file in the main application window.

The following device properties are displayed:

Column	Description
Device	SCP device name, non-editable
Type	SCP device type, non-editable
OPC alias	OPC alias name of the device, editable, optional

Fig. 4-88: Device properties displayed

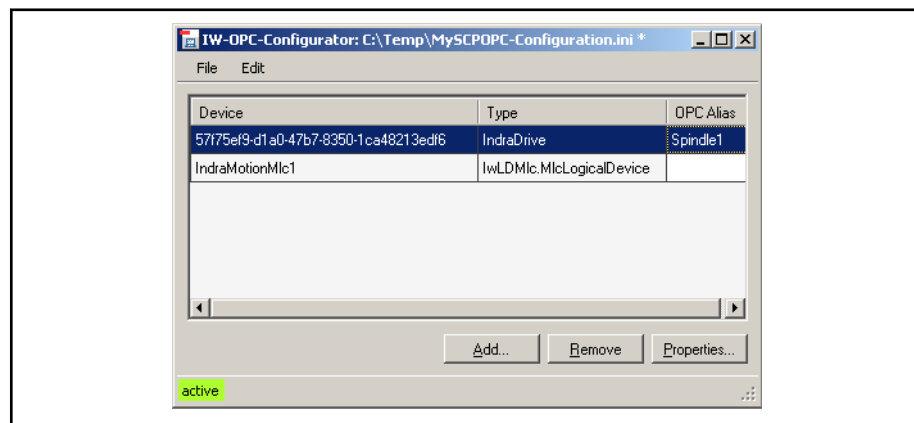


Fig. 4-89: Display of configuration file

Operation

Creating Device Configuration

Use the main menu option **File ► New**. Enter the file name when saving the configuration. If there still are unsaved configuration data when exiting the application, you will be requested to save the data.



The title bar of the application shows the path and file name of the currently opened file. If a file has been modified since it was opened or since it was last saved, the symbol "*" appears at the end of the file name in the title bar of the dialog.

Opening Device Configuration

Use the main menu option **File ► Open**. Select the directory and the desired configuration file in the file selection dialog. Only files with the file extension ".ini" are shown. The SCP-OPC configuration is displayed as a list of devices in the main application window. If the open file is the currently active SCP-OPC configuration, a light-green field with the label "active" will be displayed in the status bar of the application.



The title bar of the application shows the path and file name of the currently opened file. If a file has been modified since it was opened or since it was last saved, the symbol "*" appears at the end of the file name in the title bar of the dialog.

Adding Devices

Use the main menu option **Edit ► Add**, the context menu option **Add** or the button **Add** to display a selection dialog for selecting the devices which you would like to add to the configuration. Devices already included in the configuration will be grayed out in the selection dialog. IndraWorks projects, SCP-

OPC configuration files or an active SCP-OPC configuration are available as data sources. Using the "Properties" option, both the selection dialog and the main window of the configurator can display a dialog showing the properties of an SCP device for each device.



The devices included must be unique within a configuration. If devices are selected in the selection dialog which are already included in the configuration (shown in grey), these will not be added again.

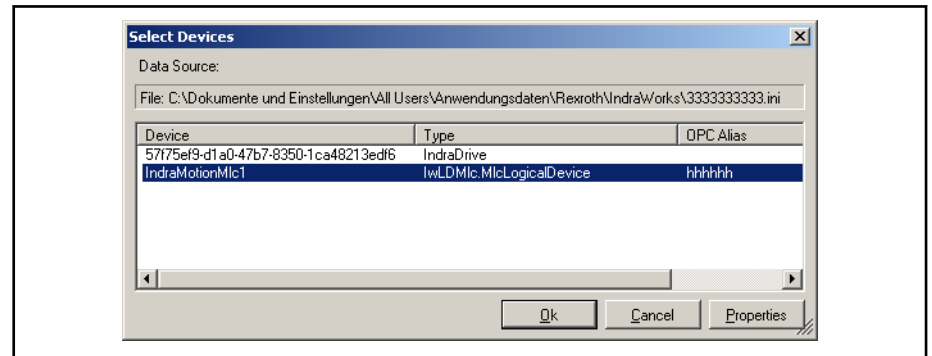


Fig. 4-90: Dialog: Select devices

Displaying Device Properties

Use the main menu option **Edit ► Properties**, the context menu option **Properties** or the button **Properties** to display the properties dialog of the selected device in the device list with all properties. The option is activated only if exactly one device on the list has been selected.



The Device Properties dialog can be called up from the main window as well as from the Select Devices dialog.

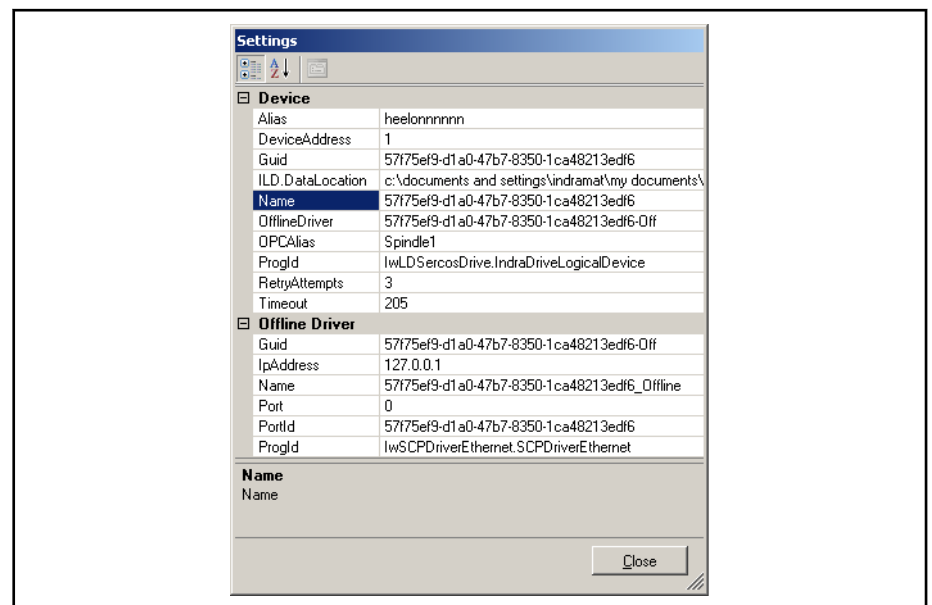


Fig. 4-91: Dialog: Settings (device settings)

Overview

Changing Device OPC Alias

Device name and device type cannot be changed by the user. These fields are displayed as non-editable fields with a grey background. You can however change the OPC alias name of a device by selecting the desired device and clicking on the OPC alias name field. Exit editing mode with <Return> or by selecting another field.



The OPC alias name may include only the following characters:

"a-z, A-Z, 0-9, _, -".

If the naming convention is violated, a message will be displayed, the background of the field will turn red and an error symbol will be displayed in the field. A description of the error that has occurred will also appear in the tool tip of the field marked as incorrect.

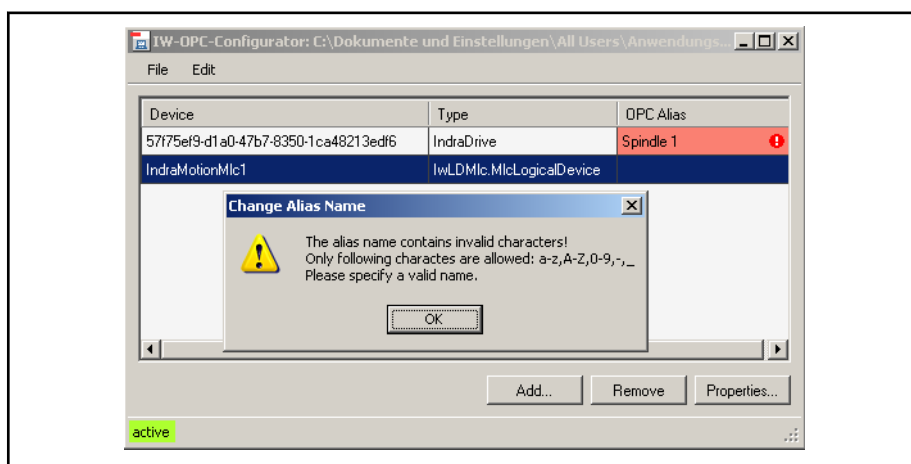


Fig.4-92: Dialog: Changing the alias name

Removing Devices

Use the main menu option **Edit ► Remove**, the context menu option **Remove** or the button **Remove** to remove the selected devices from the device list after confirming a confirmation prompt.



The option is activated only if at least one device has been selected from the list.

Saving Device Configuration

Use the main menu option **File ► Save** to save the currently edited configuration to the folder and with the file name displayed in the title bar.



This option is only available if any changes were made since the data was opened or last saved.

The "Save" option is not available for newly created configurations as long as no file name has been allocated.

Saving Device Configuration as

Use the main menu option **File ► Save As...** to save the currently edited configuration under a different file name. Select the directory you want in a file selection dialog and enter the file name under which you wish to save the file. If the file already exists in the target directory overwriting of the file has to be confirmed.



The "Save" option is not available for newly created configurations because no file name has been allocated. The option "Save As" is always available providing a configuration file is open.

Activating Device Configuration

Use the main menu option **File ► Activate** to declare that the currently opened file is to be used as the active configuration. That means that this configuration will be used to generate the device list of the OPC server when the OPC server next restarts. Only devices which are defined in this configuration file are available via the OPC server.

If the open file is the currently active SCP-OPC configuration, a light-green field with the label "active" will be displayed in the status bar of the application.



A configuration can only be designated as an active configuration if the configuration has been previously saved as a file.

Exiting the Application

Use the main menu option **Exit** or **x** in the system menu to exit the application. If there are unsaved data you will be prompted to save the data.

lwSCP and OPC.lwSCP Items

General Information

This chapter provides an overview of the items available in the "lwSCP server" which are relevant for the OPC client.

When using the items in the OPC client it should be ensured that the item syntax described for the "lwSCP server" is extended with the specification of the "requested data type" and the "control name or alias".

For direct comparison, the item syntax will be shown first for the "lwSCP" server and subsequently for the "OPC.lwSCP" server.

lwSCP Server

- PLC variables
"[lwSCP service],[fully qualifying variable name]"
Example:
"Plc.PVL,Application.HMIVARS.bLifeSign_hmi"
- Sercos parameter
 - MLC/MLP
"[lwSCP service],[drive address],[parameter number],[element coding]"
Example:
"SPA1,0,C-0-0001,D"
"SPA1,0,C-0-0001,40"
 - MLD
"[lwSCP service],[parameter number],[element coding]"
Example:
"SPA2,S-0-0030,D"
"SPA2,S-0-0030,40"

OPC.lwSCP Server

- PLC variables

Overview

"![required data type],[control name or alias],[lwSCP service],[fully qualifying variable name]"

Example:

"!Bool,ImMlc2G,Plc.PVL,Application.HMIVARS.bLifeSign_hmi"

- sercos parameter
 - MLC/MLP

"![required data type],[control name"or
"alias],[lwSCP service],[drive address],[parameter number],[element coding]"

Example:

"!UI4,ImMlc2G,SPA1,0,C-0-0001,D"

"!UI4,ImMlc2G,SPA1,0,C-0-0001,40"
 - MLD

"![required data type],[drive name or alias],[lwSCP service], [parameter number], [element coding]"

Example:

"!BSTR,Drive01,SPA2,S-0-0030,D"

"!BSTR,Drive01,SPA2,S-0-0030,40"

The "OPC.lwSCP data types" listed in the following table can be used for the parameter "required data type".

OPC.lwSCP Data types	IEC data types	Value range Size of memory
BOOL	BOOL	TRUE (1) and FALSE (0) 8 bits
I1	SINT	-128 to 127 8 bits
I2	INT	-32768 to 32767 16 bits
I4	DINT	-2147483648 to 2147483647 32 bit
I8	LINT	-2 ⁶³ to 2 ⁶³ -1 64 bits
UI1	USINT	0 to 255 8 bits
UI2	UINT	0 to 65535 16 bits

Overview

OPC.lwSCP Data types	IEC data types	Value range Size of memory
UI4	UDINT DATE TIME DATE_AND_TIME DT TIME_OF_DAY TOD	0 to 4294967295 32 bit
UI8	ULINT LTIME	0 to $2^{64}-1$ 64 bits
R4	REAL	$\pm(1.175494351e-38$ to $3.402823466e+38)$ 32 bit
R8	LREAL	$\pm(2.225073858072014e-308$ to $1.7976931348623158e+308)$ 64 bits
BSTR	STRING WSTRING	Any string with a length of 80 characters, when no length is specified in the declaration in the PLC

Fig. 4-93: OPC.lwSCP data types

IndraLogic Items

Overview

The IndraLogic items relevant for the OPC client are listed below.

Item overview

- **Plc.State**
Reading current status information of the IndraLogic Runtime.
- **Plc.PVT**
Reads symbol information from the PLC project.
- **Plc.PVF**
Reads symbol information of a variable.
- **Plc.PVL**
Reads and writes variable values.
- **Plc.PVBS**
Access to PLC variables (structures, arrays) as a byte stream
- **Plc.PVLK**
Access to PLC variables (structures, arrays)

sercos Items

Overview

The sercos items relevant for the OPC client are listed below.

Item overview

- **SPH1**
Reads and writes the sercos phase.

Overview

- **SPA1**
Reads and writes elements of the sercos parameters.
- **SPA2**
Reads and writes elements of the sercos parameters of a drive.

Disclaimer

WARNING

No liability for damages caused by customer OPC clients

Via OPC, you have unlimited access to all device data and the entire device functionality.

For damages and loss of data due to improper OPC handling in "Third Party" OPC client applications and WinStudio operating screens or due to a non-compliance of device-specific safety guidelines, Bosch Rexroth does not assume liability. This affects writing of device-sensitive and safety-relevant data in particular.

4.4.3 BRWS Driver (Bosch Rexroth WinStudio Driver)

Properties

The driver integrated in WinStudio from WinStudio version 6.2 offers the possibility to access variables and messages of the XLC/MLC systems on EtherNet basis for PC-based and embedded PC-based visualization devices.

The driver uses the communication server to access variables. The BRWS driver can communicate with a maximum of 20 different controls (IP addresses) at the same time.

The BRWS protocol only uses synchronous data services. The client sends a request and the server immediately sends a response.

Application Area

Communication of the control with IndraWorks visualization projects on PC-based visualization devices (VSx.3 and VPx.3)

Communication of the control with IndraWorks visualization projects on embedded PC-based (XPe) visualization devices (VEPxx.4 and VEHP.2)

Communication of the control with WinStudio stand-alone visualization projects on PC-based visualization devices (VSx.3 and VPx.3)

Communication of the control with WinStudio stand-alone visualization projects on embedded PC-based (XPe) visualization devices (VEPxx.4 and VEHP.2)

Communication of the control with WinStudio stand-alone visualization projects on embedded PC-based (CE6.0) visualization devices (VEPxx.4)

Configuration

The BRWS driver is configured in the WinStudio development environment.

Open the context menu of the communication node in the IndraWorks Project Explorer and select **Insert Driver**. The driver (BRWS connection with communication server) can be selected in the dialog of the communication driver. The driver is then displayed in the lower dialog window.

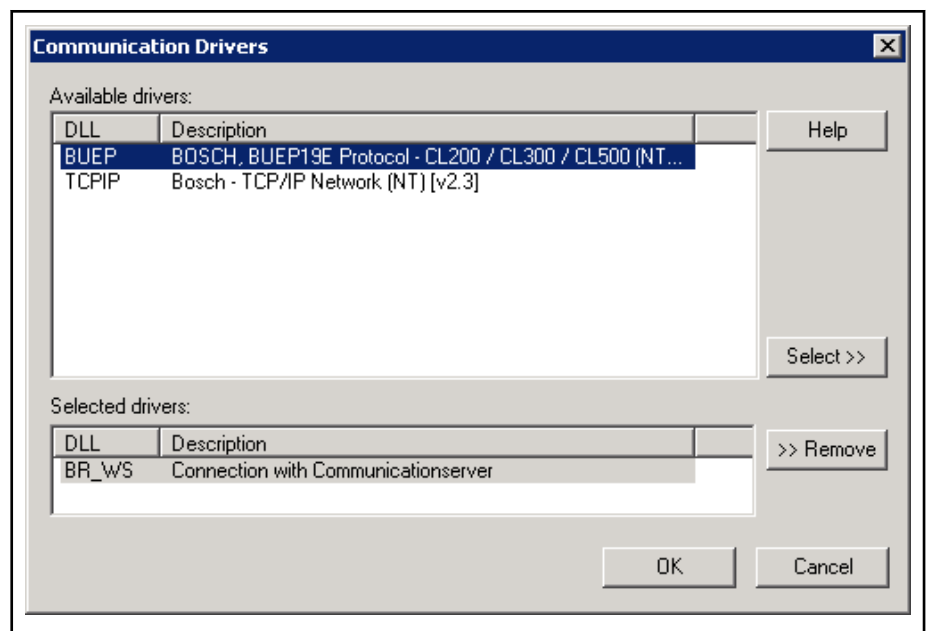


Fig. 4-94: Driver selection

After confirming with **OK**, a new folder "BRWS" is created in the WinStudio operating range. The driver can be configured and new driver sheets can be created.

The TCP/IP address is specified as target control in the driver settings dialog. Select "None" as "Serial Encapsulation".

A timeout can be specified in milliseconds in the field "timeout". The set time is used for timer monitoring of the communication connections. The timeout is only important if the visualization is connected to two or more controls and in case one control fails or cannot be reached.

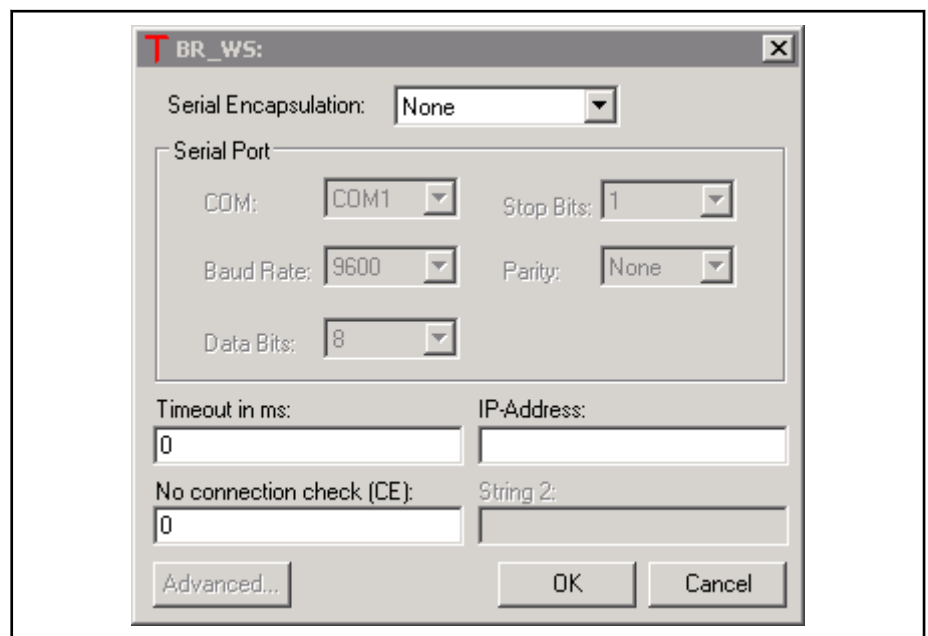


Fig. 4-95: Driver properties

Overview



In case of bad LAN connections, network failures can occur in XP devices which are logged with error number 1005. In this case, the timeout has to be increased to 300 - 500 ms!

If multiple CE devices communicate with the same control, temporary power failures (1005) occur to relieve the communication channel if the connection check is to be switched off.

Recommendation: If projects with large symbol tables are loaded to the control, e.g. during the configuration phase by means of "Online Change", a timeout of at least 1000 to 2000 ms is recommended!

Driver Sheets

By means of the driver sheets, the communication can be defined in more detail.

By right-clicking on the BRWS folder, new driver sheets can be created. They are displayed as a list below the folder and can also be selected and changed.

The driver sheets are divided into two parts:

- **Header:**

The points of time and the priorities to read or write are specified in the header.

- **List area:**

The relation of variables in the applications to variables in the control are defined in the list area.

Description: ☐ Increase priority

Read Trigger: Enable Read when Idle: Read Completed: Read Status:

Write Trigger: Enable Write on Tag Change: Write Completed: Write Status:

Station: Header: ☐ Min:
☐ Max:

	Tag Name	Address	Div	Add
*				
*				
*				
*				
*				

Fig.4-96: BRWS driver sheet

Header of driver sheets:

- **Description**

Description of the driver sheet for documentation purposes.

- **Check box: Increases the priority**

Enabling the check box increases the priority for reading and writing of the driver sheet variables.

- **Read trigger**

Driver sheet variables are read when changing the entered tag.

- **Reading when idle**

If a constant or a tag unequal to zero is entered in this field, the driver sheet variables are read continuously.

- **Reading ended**

The entered tag is set if the read command has been completed.

- **Read status**

The status of the last read command is transmitted to the entered tag.

- **Write trigger**

Driver sheet variables are written when changing the entered tag.

- **Write on change**

If a constant or a tag unequal to zero is entered in this field, the driver sheet variables are continuously checked for changes. Upon a change, the driver sheet variables are written.

- **Writing completed**

The entered tag is set if the write command has been completed.

- **Write status**

The status of the last write command is transmitted to the entered tag.

- **Station**

This field offers more driver option for the communication with other controls.

If it is left blank, the IP address of the previously set target control is used to read/write data (see [fig.4-95 Driver properties, page 93](#)).

If the IP address of a control is entered in the station, communication takes place with this control. The IP address can be changed during runtime. This means that the currently entered IP address is always selected. The accessed controls need to have loaded the used symbols and should also be accessible physically via the TCP/IP, otherwise no communication can be established.

It is thus possible to use the selected visualization on different controls without having to adjust the project.

- **Header**

A unique ID has to be entered in the header.

The ID structure is irrelevant. Only numbers and characters can be used. The ID has to be unambiguous, i.e. an ID should only be assigned once in a project. If the number of the driver sheet is used in the ID, no duplicate IDs can be assigned.



Avoid simultaneous "Enable read when idle" and "Enable write on tag change", if possible! Variable reading should take place via "Read trigger".

List area of the driver sheets:

- **Variable name**

Overview

The WinStudio variables that have to be written with the variable value of the address column (reading) or whose value has to be written to the variable in the address column (writing) are entered here.

- **Address**

The control variable name and type is entered in this field

The variable name consists of <Program block name>.<Variable name>;<Type>

- **Example:** "Global Variable"
.Glob_int;T_INT16
.Glob_write;T_UTF8STR
- **Example:** "Local variable in program block PLC_PRG"
PLC_PRG.Byt;T_INT8
PLC_PRG.Ind;T_INT16

Protocol Structure

The request length in UTF-8 format precedes each protocol. The request length is not identical with the data security layer (transport/network), if the request is fragmented during the transport and/or several requests are transmitted simultaneously.

Header	Body
UTF-8 data length 0-7FFF FFFF (1-6Byte)	Requests/responses

Volume of Data

General driver sheet information:

- max. 512 items per driver sheet
- no blank lines in the driver sheet
- all variables have to be in the control
- case sensitivity in the address fields has to be identical to the notation in the symbol file
- max. 960 characters per address field

Information about EIS:

- The number of the handles available for communication is limited to 10000 in case of EIS (DeviceXCC.xml or DevicePlc.cfg).
- The number of the required handles is composed as follows:
Each registered PLC variable requires a handle. Additionally, several group handles can be registered for each driver sheet.
- The TCP window size in MLC BSP is set to 1460 bytes (1 telegram). This means that if the data volume of a driver sheet is greater than 1460 bytes, it has to be partitioned which results in performance loss.

PLC Variable and Parameter Support

PLC Variables

- Item syntax
 - 'POE_NAMESPACE'.PlcVariable';BRWS_DataType'

PLC data type	BRWS data types	WinStudio
BOOL	T_BOOL	Boolean
BYTE	T_UINT8	integer
WORD	T_UINT16	integer
DWORD	T_UINT32	integer
LWORD	T_UINT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".
SINT	T_INT8	integer
INT	T_INT16	integer
DINT	T_INT32	integer
LINT	T_INT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".
USINT	T_UINT8	integer
UINT	T_UINT16	integer
UDINT	T_UINT32	integer
ULINT	T_UINT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".
REAL	T_FLOAT32	real
LREAL	T_FLOAT64	real
STRING	T_UTF8STR	String
WSTRING	T_UTF8STR	String
TIME	T_UINT32	integer
LTIME	T_UTF8STR	String
DATE	T_UINT32	integer
DATE_AND_TIME	T_UINT32	integer
TIME_OF_DAY	T_UINT32	integer

Fig.4-97: PLC data type assignment



Converting to a data type with a greater value range is possible.
E.g. "T_UINT8 ⇒ T_UINT16 ⇒ T_INT32 ⇒ T_UTF8STR"

XLC/MLC Parameters

- Item syntax
 - 'CtrlNo','AxisNo','SercosParameter'. 'SercosElement'; 'BRWS_Data-Type'

Overview

BRWS item element	Value	Description	
CtrlNo	0	Local control	Optional
	1 - 999	Control address in the link	
AxisNo	0	Control parameter	
	1 - 999	Axis parameter	
SercosParameter	C	System parameter	
	K	Kinematics	
	M	Touch probe	
	N	Cam switch	
	O	Oscilloscope	
	A	Axis-specific parameter	
	S	Standard drive parameter	
	P	Product drive parameter	
SercosElement	'Empty'	Data	
	1	Sercos ID	
	2	Description	
	3	Attributes	
	4	Unit	
	5	Min. value	
	6	Max. value	
	7	Data	
	8	Data status (only drives)	

Fig.4-98: BRWS item structure for parameters

Parameter type	PLC data type	BRWS data types	WinStudio
BINARY, 2BYTE	WORD	T_UINT16	integer
UINT_DEC, 2BYTE	UINT	T_UINT16	integer
INT_DEC, 2BYTE	INT	T_INT16	integer
UINT_HEX, 2BYTE		T_UINT16	integer
IDN, 2BYTE		T_UINT16	integer
BINARY, 4BYTE	DWORD	T_UINT32	integer
UINT_DEC, 4BYTE	UDINT	T_UINT32	integer
INT_DEC, 4BYTE		T_INT32	integer
UINT_HEX, 4BYTE	UDINT	T_UINT32	integer
MLC_IDN, 4BYTE		T_UINT32	integer
FLOAT, 4BYTE	REAL	T_FLOAT32	real
TEXT, 1BYTE_VAR	STRING	T_UTF8STR	String

Overview

BINARY, 8BYTE	LWORD	T_UINT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".
UINT_HEX, 8BYTE	ULINT	T_UINT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".
DOUBLE, 8BYTE	LREAL	T_FLOAT64	real
MLC_IDN, 8BYTE	MB_IDN (LWORD)	T_UINT64	"not supported" Configure variable with BRWS data type "T_UTF8STR".

Fig.4-99: Parameter data type assignment



Converting to a data type with a greater value range is possible.
E.g. "T_UINT8 ⇒ T_UINT16 ⇒ T_INT32 ⇒ T_UTF8STR"

Error Messages

The communication state can be monitored or displayed in several ways.

The status can be queried and displayed for each driver sheet on the CE target. This status shows, whether the communication of the driver sheet with the control was successful or not. Writing and reading accesses are considered separately.

The communication status can be monitored and/or written in a log file using the development environment of WinStudio via "Remote LogWin".

The following messages are output by the Bosch Rexroth WinStudio driver:

-
- 0 ; OK
 - 1003 ; No IP address available for this data source
 - 1004 ; Com server login failed
 - 1005 ; Communication error
(Data source not available, connection to control configured incorrectly, no physical connection to the control, network problems etc.)
 - 1006 ; Time-out of reply to a query at Com server (> 30 seconds)
 - 1008 ; Invalid handle
(Variable in driver sheet not in control symbols, symbol file loaded?, reload program, incorrect symbol file)

Fig.4-100: Messages of the Bosch Rexroth WinStudio driver

Information about the cause are already contained in the messages.

Message Information

- 0 No error. Communication is online.
- 1003 The IP address is entered in the target control during driver configuration. If this entry is blank or incorrect, the driver cannot connect to the communication server on the target control.

Overview

- 1004** Com server login failed. The basic control connection was established, however, the login request response is negative or there is no response at all. Either none or an old status of the Com server is installed in the control or the control is in an undefined state and the com server task cannot be executed anymore.
- 1005** The communication error is always displayed if sending and receiving of the com server data failed.
- 1006** The com server does not react to client requests. The control is busy or an exception has been reported.
- 1008** Invalid handle. This message is send by the communication server if a variable which does not exist in the control is requested by the client. This should not occur during normal operation, as all variables on the driver sheet have a counterpart in the control in case of correctly filled driver sheets (verify using the "Check" function of the browser).
- If the program is reloaded, the communication server cancels all previously registered handles. By means of feedback 1008 of the communication server, the client can detect that its handle is not valid anymore. The communication is re-established by the client. All handles are registered again. When reloading the program, the client reports an 1008 error for several seconds until the client is synchronized again.
- If the "1008" error is continuously present, there are three causes of defects:*
1. At least one driver sheet was not correctly filled out and contains variable descriptions which are not contained in the symbol file of the control.
Solution: Check driver sheets using the "Check" function of the browser.
 2. An incorrect or incomplete symbol file was loaded in the control.
Solution: Check, if the project download was executed with symbol information and if the correct symbol file has been loaded.
 3. The com server does not have enough handles.

Documentation

The following documentation is available for a detailed description of the BRWS driver (Bosch Rexroth WinStudio driver):

DOK-CONTRL-IL2GPRO*Vxx-APyy-EN-P

xx - Version

yy - Edition

4.4.4 IndraLogic OPC.DA

Properties

The IndraLogic OPC server OPC.DA is an OPC server to access the PLC system of the control.

IndraLogic OPC.DA can be used for the PC-based and embedded PC-based visualizations (see [chapter 4.2.1 overview, page 27](#)).

The server always has to be configured to operate an IndraLogic OPC.DA server. It is specified with which devices the OPC.DA server connects upon the start (see [chapter Configuration of the IndraLogic OPC.DA server, page 101](#)).



Detailed information about the OPC communication, the items that can be used and a possibly required Client programming are available in the DOK-IM*ML*-OPC*COM*Vxx-APxx-EN-P documentation

Configuring the IndraLogic OPC.DA Server

General Information

This chapter describes the configuration of the IndraLogic OPC.DA server "IndraLogic.OPC.DA" for PC systems (Windows XP and newer operating systems) and CE systems (Windows CE 6) for communication with IndraLogic 2G control systems.

The OPC.DA server is configured using the "OPCServer.ini" file in the "%CommonAppData%\Rexroth\IndraWorks" directory:

Example:

Paths:

"C:\Documents and Settings\All Users\Application Data\Rexroth\IndraWorks" on PC systems or in the "\StorageCard\Rexroth\Controls\OPCServer" directory of CE systems.

The "OPCServer.ini" is created on the PC-based control panel using a configurator. See also [chapter "OPC.DA Server Configuration" on page 101](#).

The manual configuration of the "OPCServer.ini" file (e.g. for WindowsCE-based visualization devices) is described under section [chapter "OPC.DA Server Configuration File OPCServer.ini" on page 105](#).



A symbol configuration must be present on the control to enable communication with the control. See [chapter 5.2 Symbol Configuration, page 129](#).



The IndraLogic OPC.DA server supports the communication to the IndraLogic 2G control systems.

This chapter describes only the relevant entries in the OPC.DA server configuration file "OPCServer.ini" for communication with IndraLogic 2G control systems.



The communication of the OPC.DA server with the IndraLogic 2G control systems is only possible with the visualization devices "VEPxx.4".

OPC.DA Server Configuration

Identifying the Communication Address

For communication with the control, the OPC.DA server or the underlying communication component "GatewayService" require a "communication address" which is determined from the IP address and the SubNet mask of the control.

In order to exclude the communication address as a source of error in the configuration file it always has to be identified by carrying out a communication test with the control.

The following figure shows how the communication address "000.c0a8.011f" was identified for the control system IndraMotion MLC with IP address "192.168.1.31". After a successful communication test the communication address is displayed in the line "Address:" in the section "PLC communication successful:" of the field "Result of connection test" as a hexadecimal number and used in this form for configuring the OPC.DA server.

Overview

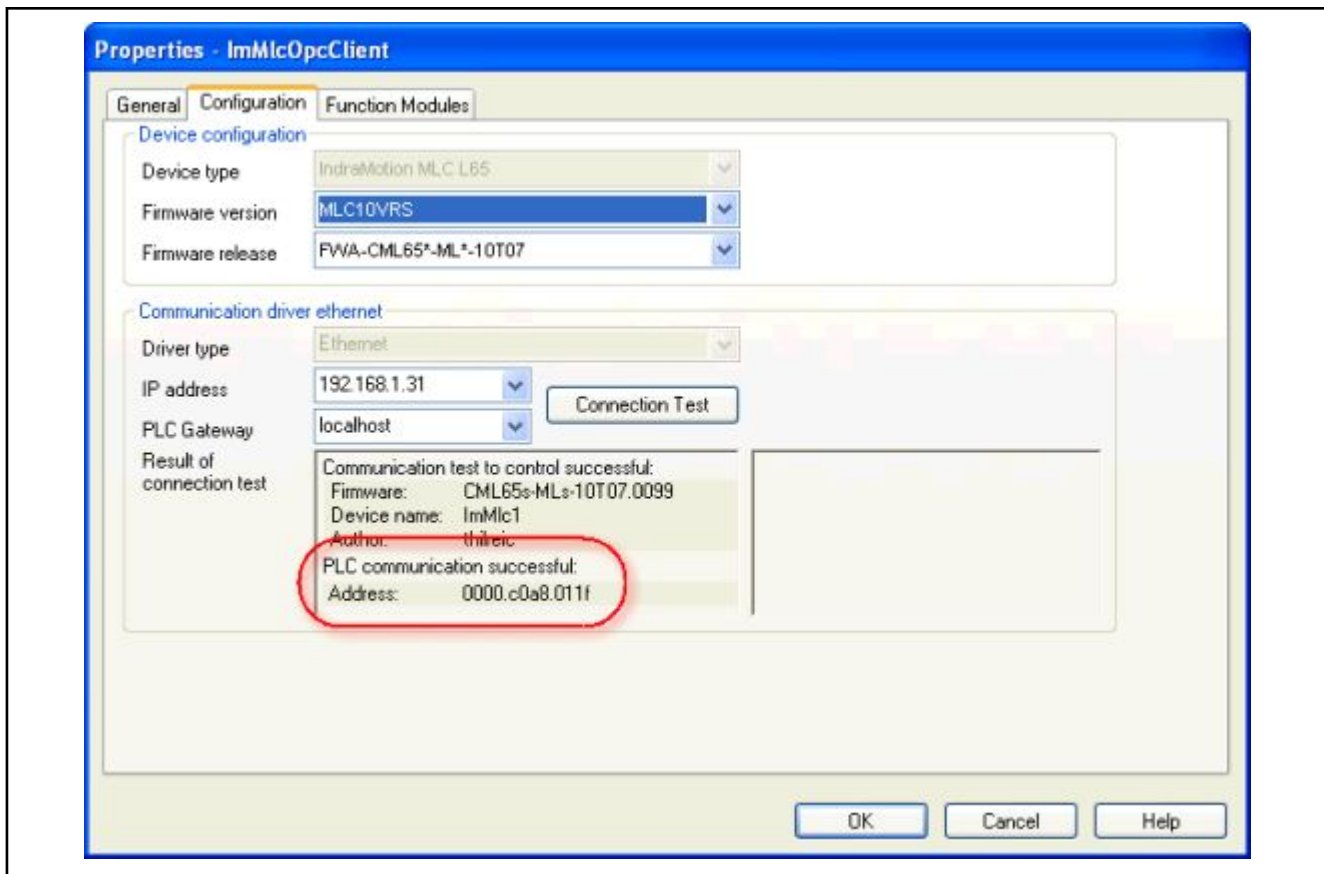


Fig.4-101: Properties dialog IndraMotion MLC - Identifying communication address

OPC.DA Server Configurator

A configurator is available for PC-based visualization devices for the configuration of the OPC.DA server

The call is executed via the Windows start menu of the installed IndraWorks version

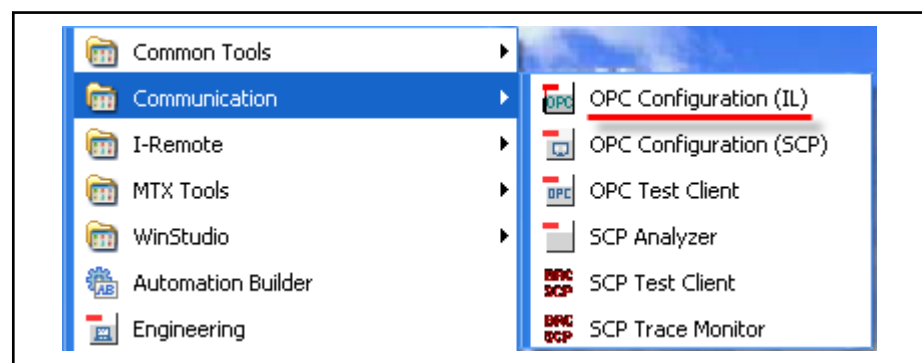


Fig.4-102: OPC.DA server configurator call

When creating a new configuration, the call is send via the context menu and by attaching a new PLC.

- Logging should be disabled for the "regular operation" as it can only be used during the commissioning.

- The standard update rate of 200ms is recommended for registering groups and items at the OPC.DA server. Higher update rates are not required in most cases.



A detailed description of the configuration parameters can be found under [chapter "OPC.DA Server Configuration File OPC-Server.ini"](#) on page 105

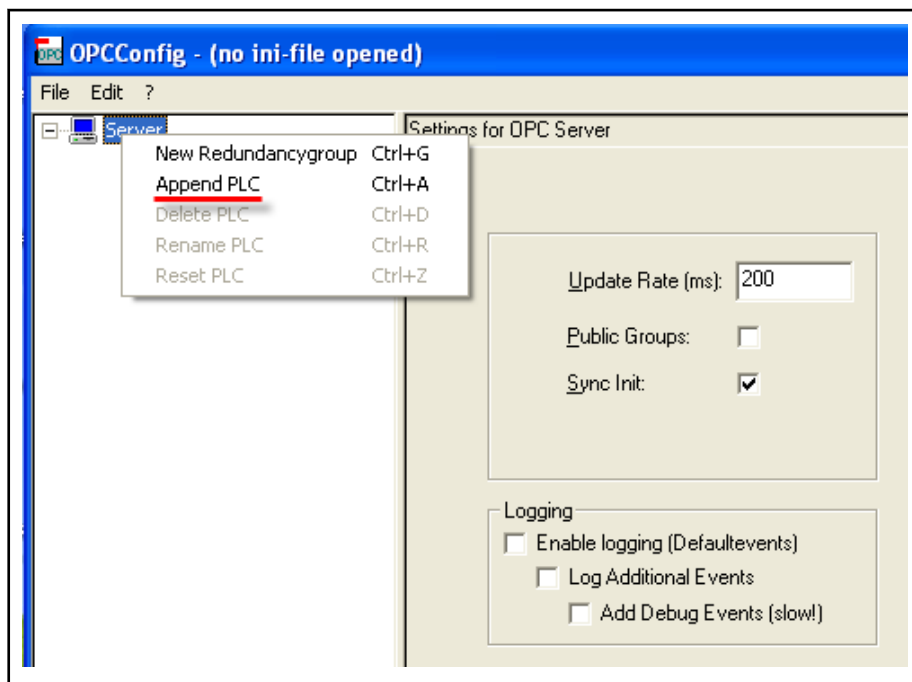


Fig. 4-103: Creating a new control within the OPC.DA server configuration

For the communication of the OPC.DA server with 2G control systems, an interface "**GATEWAY3**" (= default) has to be selected over which the connection to the control can be established

Overview

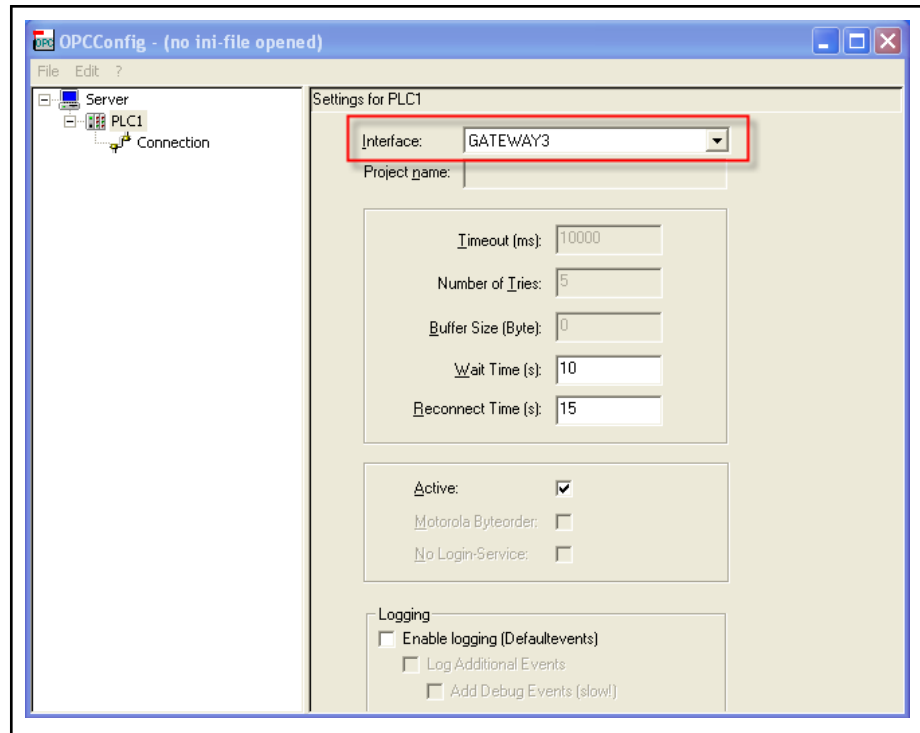


Fig.4-104: Configuration of the interfaces (gateway) for the communication in the OPC.DA server configurator

Subsequently, the connection settings have to be selected where the IP address of the 2G control has to be entered in the HEX.

The address as determined under [chapter "Identifying the Communication Address" on page 101](#) in IndraWorks Engineering was used in the case described.

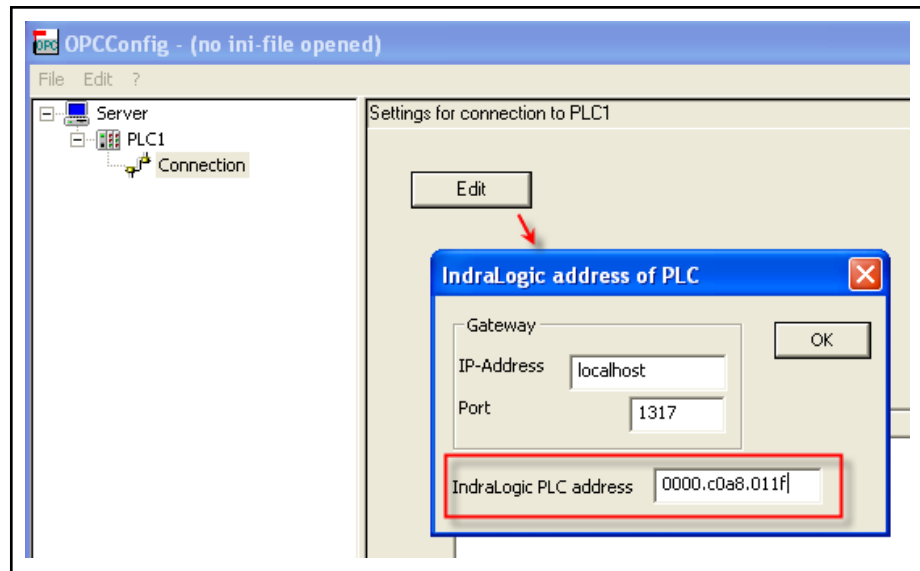


Fig.4-105: Configuration of the connection settings in the OPC.DA server configurator

The configuration has to be saved under "OPCServer.ini" to become active upon the next OPC.DA server start

For configuration file paths, see [chapter "General Information" on page 101](#)

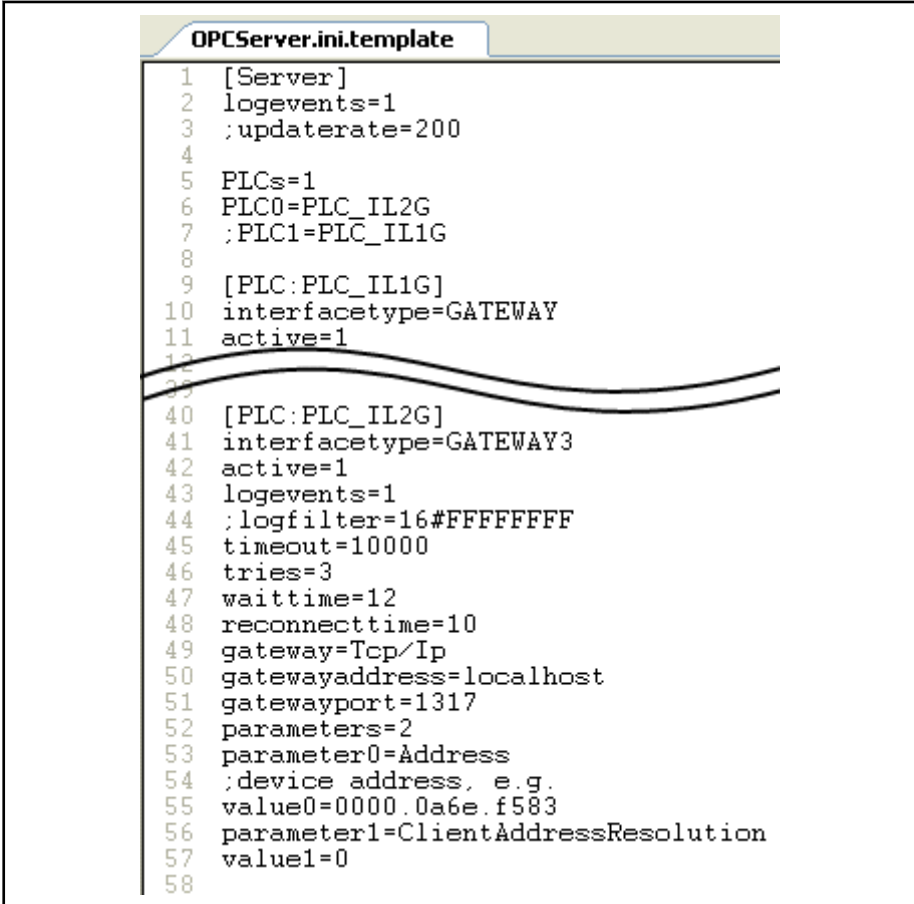
OPC.DA Server Configuration File "OPCServer.ini"

The OPC.DA server configuration file "OPCServer.ini" is structured like an "INI file", i.e. it contains sections and parameter with names and values. The character ";" (semicolon) is used to insert comment lines for documentation purposes or to mask parameter.

The template file "OPCServer.ini.template" (fig. 4-106 "Configuration file template of the IndraLogic OPC.DA server - OPCServer.ini.template" on page 105) can be used to create an OPC.DA server configuration. The template file is stored with every IndraWorks Suite "Engineering" or "Communication" installation version in the directory "%CommonProgramFiles%\Rexroth".

The placeholder or the environment variable "%CommonProgramFiles%" is set by the operating system depending on the installed operating system language and can be queried by entering the line "echo %CommonProgramFiles%" in a prompt. In case of English interfaces or MUI (Multilingual User Interface) operating systems, the path "C:\Program Files\Common Files" is output.

Section [chapter "Windows XP" on page 109](#) for PC systems and section [chapter "Windows CE" on page 109](#) for CE systems show in which directory the configuration file "OPCServer.ini" for the OPC.DA server has to be stored.



```

OPCServer.ini.template
1  [Server]
2  logevents=1
3  ;updaterate=200
4
5  PLCs=1
6  PLC0=PLC_IL2G
7  ;PLC1=PLC_IL1G
8
9  [PLC:PLC_IL1G]
10 interfacetype=GATEWAY
11 active=1
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40 [PLC:PLC_IL2G]
41 interfacetype=GATEWAY3
42 active=1
43 logevents=1
44 ;logfilter=16#FFFFFFFF
45 timeout=10000
46 tries=3
47 waittime=12
48 reconnecttime=10
49 gateway=Tcp/Ip
50 gatewayaddress=localhost
51 gatewayport=1317
52 parameters=2
53 parameter0=Address
54 ;device address, e.g.
55 value0=0000.0a6e.f583
56 parameter1=ClientAddressResolution
57 value1=0
58
  
```

Fig. 4-106: Configuration file template of the IndraLogic OPC.DA server - "OPCServer.ini.template"

The following tables [fig. 4-107 "OPC.DA server configuration file - server section \[server\]" on page 106](#) and [fig. 4-108 "OPC.DA server configuration file -](#)

Overview

[control section \[PLC:...\]](#) on page 107 show the configuration parameter for the server and control section of the file "OPCServer.ini".

Parameter name and parameter value	Description
[Server]	Section identifier of the "server" section
updaterate=200	Update rate of the OPC.DA server in milliseconds. This is the interval for updating the internal OPC cache. The setting for the OPC client update is determined by the properties of the OPC group or subscription. Depending on the number of registered variables it is necessary to change the update rate in order to reserve enough resources for other processes running on the PC. Also the update interval should not fall below 200 ms.
publicgroups=0	Creating public groups for each POU and global variable list. Public groups can be used by any client. With "public groups" it is not possible to structure variables in reference to the application (e.g. image dependent, functional groupings etc.).
logevents=0	Enabling the log mechanism. All server events are written to the file "OPCServer.log" according to the settings of the parameter "logfilter". The log file is located in the same directory as the file "OPCServer.ini". During installation or test the parameter "logevents" can be set to "1". Logging has to be deactivated again for operation, i.e. "0".
logfilter=-49	Filter definition for server events. <i>Possible values are:</i> • 0xFFFF FFFF FFFF FFFF: all (Debug) • 0xFFFF FFFF FFFF FFCE: extended logging • 0x0000 0000 0000 000A: standard logging • 0x0000 0000 0000 0000: no logging
PLCs=1	Number of configured controls.
PLC0=PLC_IL2G	List of all configured controls in the configuration file.
PLC...	Please note that the counter index (parameter name) is continuous and that a unique control section exists for all listed controls (parameter value e.g. "PLC_IL2G")

Fig. 4-107: OPC.DA server configuration file - "server" section [server]

Parameter name and parameter value	Description
[PLC:PLC_IL2G]	Section identifier of the control section "PLC_IL2G". If a connection to this control is to be established when starting the OPC.DA server, it has to be available as a value in the list of all controls of the server section in one of the parameter "PLC..." (e.g. "PLC0=PLC_IL2G").
interfacetype=GATEWAY3	Definition of control type. This parameter defines the control system that is being used for communication. <i>The following list shows the most important values.</i> • IndraLogic 2G control systems – GATEWAY3 • IndraLogic 1.x control systems – ARTI – GATEWAY (only for PC systems)
waittime=12	Waiting time in seconds of the subordinate communication layer "PLC Handler". This is the maximum waiting time to establish a connection to the control.
timeout=10000	Time in milliseconds while the server waits for an answer telegram from the control until the request is repeated. The request is repeated for as long as the parameter "tries" indicates. Only then will a connection interruption be diagnosed, i.e. after a maximum time of "timeout * tries".
tries=3	Number of repetitions for a request which is not being processed by the control. After the requests have been repeated according to the value a connection interruption is diagnosed, i.e. after a maximum time of "timeout * tries".
reconnecttime=10	Time interval in seconds during which an attempt is made to re-establish the connection.
active=1	Activate control. This parameter can be used to activate or deactivate communication with the control. e.g. the configuration of an installation can be created and modules which do not yet exist can be deactivated.
logevents=0	Enabling the log mechanism. All server events are written to the file "OPCServer.log" according to the settings of the parameter "logfilter". The log file is located in the same directory as the file "OPCServer.ini". During installation or test the parameter "logevents" can be set to "1". Logging has to be deactivated again for operation, i.e. "0".
logfilter=-49	Filter definition for server events. <i>Possible values are:</i> • 0xFFFF FFFF FFFF FFFF: all (Debug) • 0xFFFF FFFF FFFF FFCE: extended logging • 0x0000 0000 0000 000A: standard logging • 0x0000 0000 0000 0000: no logging

Overview

Parameter name and parameter value	Description
gateway=Tcp/Ip	Communication settings from OPC.DA server to "GatewayService" • gateway – Protocol used for communication. • gatewayaddress – IP4 address or host name to the gateway establishing the connection to the control. • gatewayport – Network port where the "GatewayService" accepts incoming connections. This setting must be correspond to the gateway configuration.
gatewayaddress=localhost	
gatewayport=1317	
parameters=1	Setting for the connection from GatewayService to control.
parameter0=Address	The communication address is accepted for the parameter "value0" as described in section chapter "Identifying the Communication Address" on page 101 .
value0=0000.c0a8.011f	

Fig. 4-108: OPC.DA server configuration file - "control" section [PLC:...]

fig. 4-109 "Configuration file of the IndraLogic OPC.DA server - OPCServer.ini" on page 108 shows an IndraLogic OPC.DA server configuration file establishing a communication connection on PC and CE systems with control IP address "192.168.1.31".



Fig. 4-109: Configuration file of the IndraLogic OPC.DA server - "OPCServer.ini"



The OPC.DA server configuration file "OPCServer.ini" is the same for PC and CE systems in connection with IndraLogic 2G control systems.

File Location of the OPC.DA Server Configuration File

General Information



The OPC.DA server configuration file "OPCServer.ini" of Windows XP and Windows CE are only compatible from IndraWorks version 12.

Windows XP

The OPC.DA server configuration file "OPCServer.ini" has to be contained in the

C:\Documents and Settings\All Users\Application Data\Rexroth\IndraWorks in case of an English or MUI operating system installations.

In the following graphics, the file directory for an operating system installation:

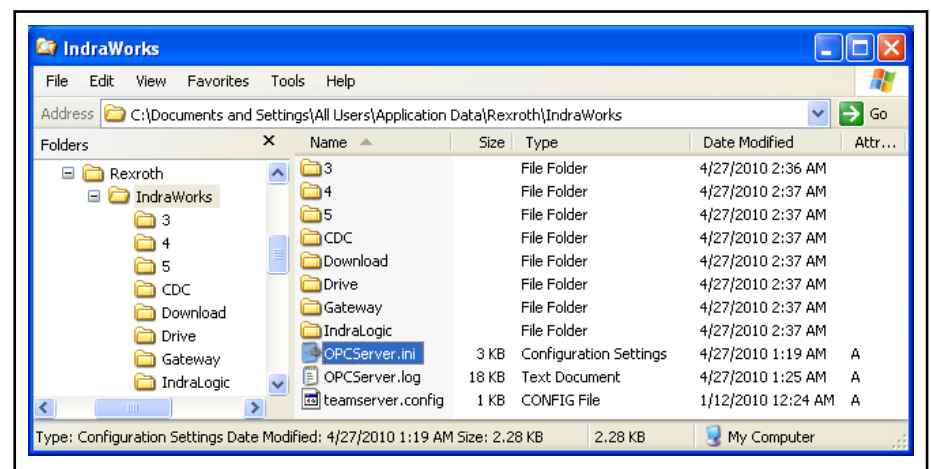


Fig.4-110: File location of the configuration file "OPCServer.ini" on PC systems

Windows CE

For Windows CE devices the OPC.DA server configuration file "OPCServer.ini" for the visualization device "VEPxx.4" has to be stored in the directory \StorageCard\Rexroth\Controls\OPCServer

Figures [fig. 4-111 "File location of the configuration file OPCServer.ini on CE systems - CE Explorer"](#) on page 110 and [fig. 4-112 "File location of the configuration file OPCServer.ini on CE systems - XP Explorer via FTP connection"](#) on page 110 show the file directory with "CE Windows Explorer" and "XP Windows Explorer" for a "VEPxx.4" visualization device.

Overview

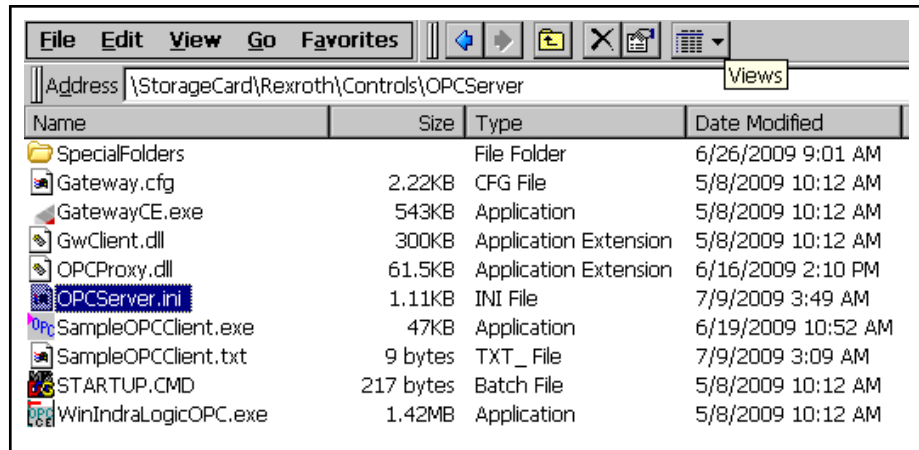


Fig.4-111: File location of the configuration file "OPCServer.ini" on CE systems - CE Explorer

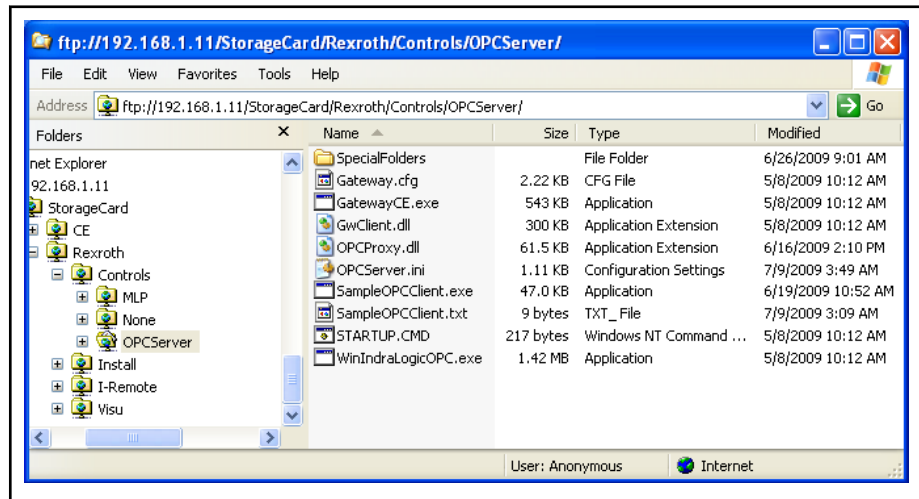


Fig.4-112: File location of the configuration file "OPCServer.ini" on CE systems - XP Explorer via FTP connection

Select "OPC.DA server" in "CE Rexroth Settings" (link "RxSettings") in the grouping field "Control" of the tab "Application Settings" to activate the OPC.DA server communication. The selection in the grouping "Visualization" is made depending on the visualization system. Refer to [fig. 4-113 "Dialog: CE Rexroth settings \(configuration of an OPC communication\)"](#) on page 110.

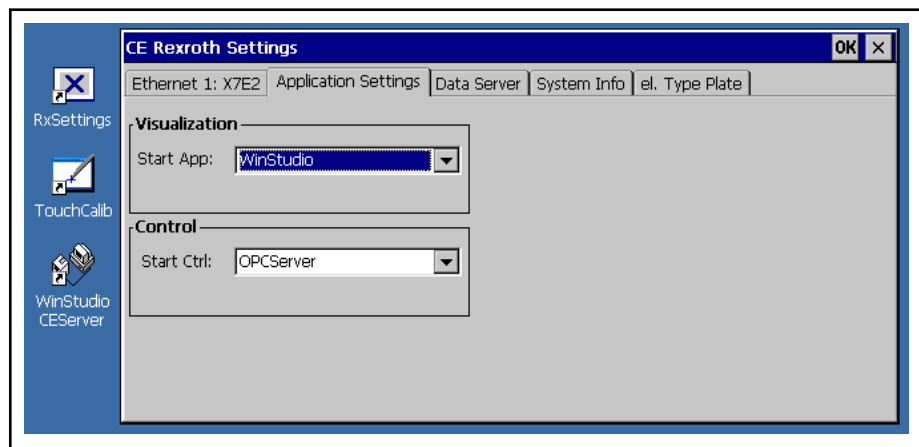


Fig.4-113: Dialog: CE Rexroth settings (configuration of an OPC communication)

OPC.DA Server - Item Syntax

The IndraLogic OPC.DA server has access to all PLC variables that were configured in the IndraWorks project for access in the symbol configuration or via pragmas. The item syntax consists of the "control name" (configuration file "OPCServer.ini") and the "fully qualifying variable name". That means the variable name is preceded by all name spaces in which the variable was declared.

A point "." is used as a separator for the control name and the fully qualifying variable name.

See the following summary.

- Item syntax
 - "[Control name].[fully qualifying variable name]"

Example:

"PLC_IL2G.Application.HMIVARS.bLifeSign_hmi"



When using the IndraLogic functionality for name space replacement via pragma "{attribute 'namespace' := 'NewNamespace'}" or using the symbol configuration, access is effected via the newly defined name space.

Example:

"PLC_IL2G.NewNamespace.bLifeSign_hmi"

Disclaimer

WARNING

No liability for damages caused by customer OPC clients

Via OPC, you have unlimited access to all device data and the entire device functionality.

For damages and loss of data due to improper OPC handling in "Third Party" OPC client applications and WinStudio operating screens or due to a non-compliance of device-specific safety guidelines, Bosch Rexroth does not assume liability. This affects writing of device-sensitive and safety-relevant data in particular.

4.4.5 OPC UA

Overview

The OPC UA server is available from version 12VRS in the "IndraMotion MLC" and "IndraLogic XLC" systems as communication server to connect HMIs, SCADA and control systems.

It permits access to all PLC variables that have been released for access in the IndraLogic symbol configuration by the respective control.

The current version of the Rexroth visualization "WinStudio" does not have an OPC UA client for communication connection to the OPC UA server in the control systems. The option to connect WinStudio via the OPC UA server is provided from the next IndraWorks versions.

This chapter gives an overview of

- the OPC UA architecture
 - [chapter "Introducing the OPC Unified Architecture" on page 112](#)

Overview

- the Bosch Rexroth OPC UA server
[chapter "Introducing the OPC Unified Architecture" on page 112](#)
- creating a communication connection to the Bosch Rexroth OPC UA server
[chapter "Bosch Rexroth OPC UA Server" on page 117](#)



The software "Wireshark", version 1.6 or higher is recommended for diagnosing connection problems between an "OPC UA client" and the "Bosch Rexroth OPC UA server".

Introducing the OPC Unified Architecture

General Information

OPC UA is the further development of the OPC industry standard. The service-oriented architecture ensures platform independency, scalability and high-availability by dispensing with a DCOM basis. OPC UA ensures an uninterrupted, vertical integration from the control level up to the automation component - irrespective of the programming language or the operating system. OPC UA is a client/server system. Several clients can simultaneously access a server. A client can access several servers.

Specification Overview

OPC Unified Architecture has been published as multi-part specification in 13 parts by the OPC foundation <http://www.opcfoundation.org>:

Part	Specification	Download
Part 1	OPC UA Part 1 - Overview and Concepts 1.01 Specification	http://www.opcfoundation.org/UA/Part1
Part 2	OPC UA Part 2 - Security Model 1.01 Specification	http://www.opcfoundation.org/UA/Part2
Part 3	OPC UA Part 3 - Address Space Model 1.01 Specification	http://www.opcfoundation.org/UA/Part3
Part 4	OPC UA Part 4 - Services 1.01 Specification	http://www.opcfoundation.org/UA/Part4
Part 5	OPC UA Part 5 - Information Model 1.01 Specification	http://www.opcfoundation.org/UA/Part5
Part 6	OPC UA Part 6 - Mappings 1.00 Specification	http://www.opcfoundation.org/UA/Part6
Part 7	OPC UA Part 7 - Profiles 1.00 Specification	http://www.opcfoundation.org/UA/Part7
Part 8	OPC UA Part 8 - Data Access 1.01 Specification	http://www.opcfoundation.org/UA/Part8
Part 9	OPC UA Part 9 - Alarms and Conditions 1.00 Specification	http://www.opcfoundation.org/UA/Part9
Part 10	OPC UA Part 10 - Programs 1.00 Specification	http://www.opcfoundation.org/UA/Part10
Part 11	OPC UA Part 11 - Historical Access 1.00 Specification	http://www.opcfoundation.org/UA/Part11

Part	Specification	Download
Part 12	OPC UA Part 12 - Discovery	
Part 13	OPC UA Part 13 - Aggregates	

Service-Oriented Architecture

The service-oriented architecture (SOA) of OPC UA is divided into logical layers.

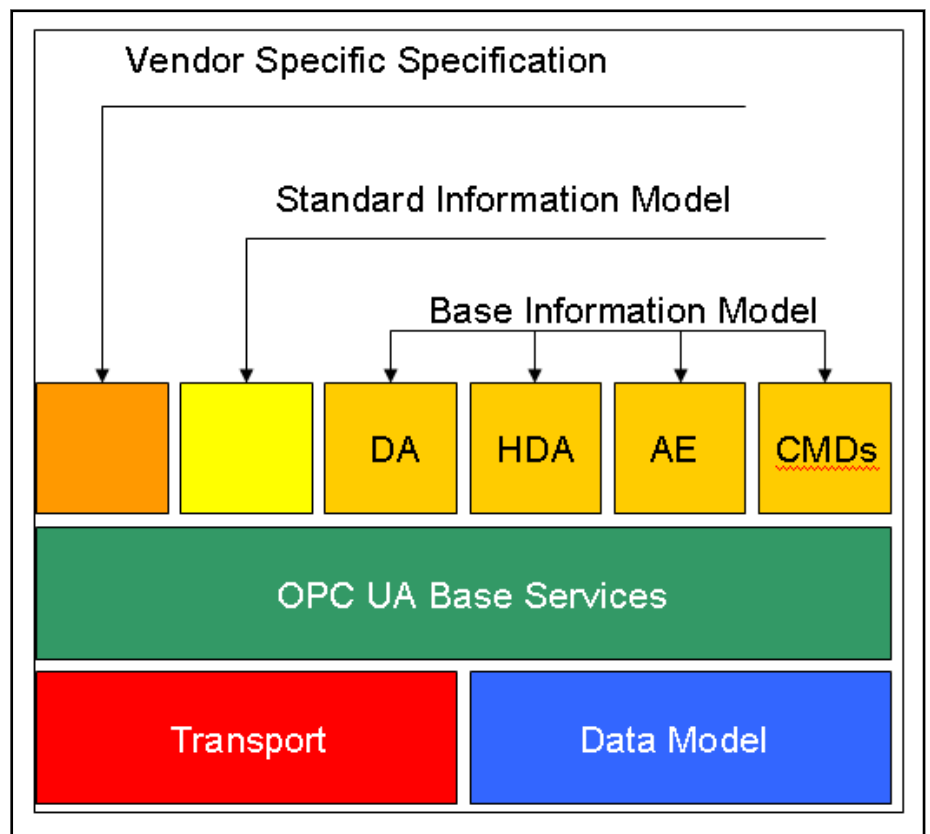


Fig. 4-114: OPC UA architecture

Transport layer

The transport layer serializes and deserializes the data and sends or receives them. The data is communicated by means of an XML stream via the HTTP-HTTPS protocol or via a high-performance binary TCP protocol.

OPC UA basic services

All OPC UA basic services are abstract method descriptions. They do not depend on the protocol and constitute the basis for all OPC UA functions. The methods are summarized in service sets. The most important sets are listed below:

Discovery Service Set

The Discovery Service Set defines services a client uses to determine the end points of a server and their security configuration. An end point is a variant, consisting of a communication protocol, a host address, a port number and security setting.

Overview

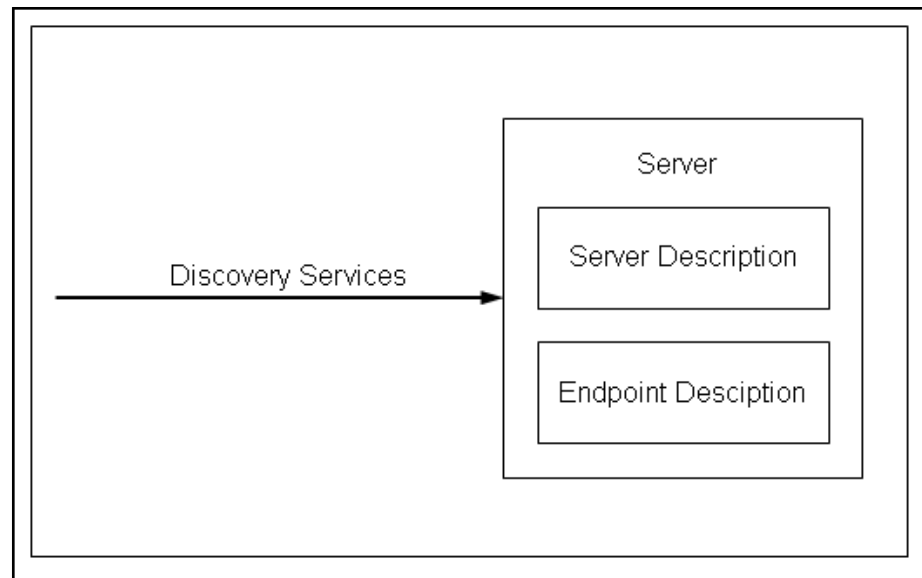


Fig.4-115: Discovery Service Set

Session Service Set

A session establishes a logic connection between client and server on an application level. The connection is established by the client and can contain client-specific user and language settings.

A session is created using the CreateSession service. The client is assigned a session ID. Via this session ID, the client can call the following data services (Read, Write, Subscribe etc.).

A session is closed automatically by the server after a client request (CloseSession) or after a timeout.

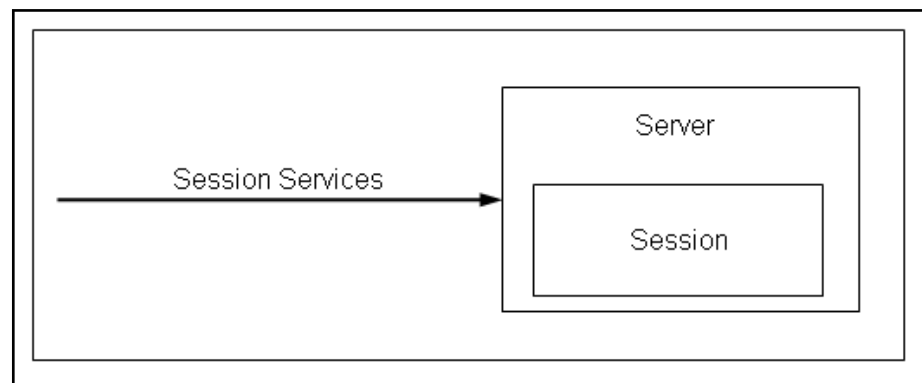


Fig.4-116: Session Service Set

View Service Set

By means of the View Service Set, the client can browse the OPC UA server address space. The address space serves to present available OPC UA server information to a client. It is generally generated by nodes. The nodes are standardized objects and serve to map "real world" objects. The most important node classes are Variable, Object and Method.

The node connections are established using references. It is differentiated between hierarchical and non-hierarchical references. Hierarchical references are used to map e.g. a structural node order. Non-hierarchical references are e.g. used to refer to a node type definition.

The type system of nodes and references can be extended by deriving existing types and can be assigned a more detailed semantics.

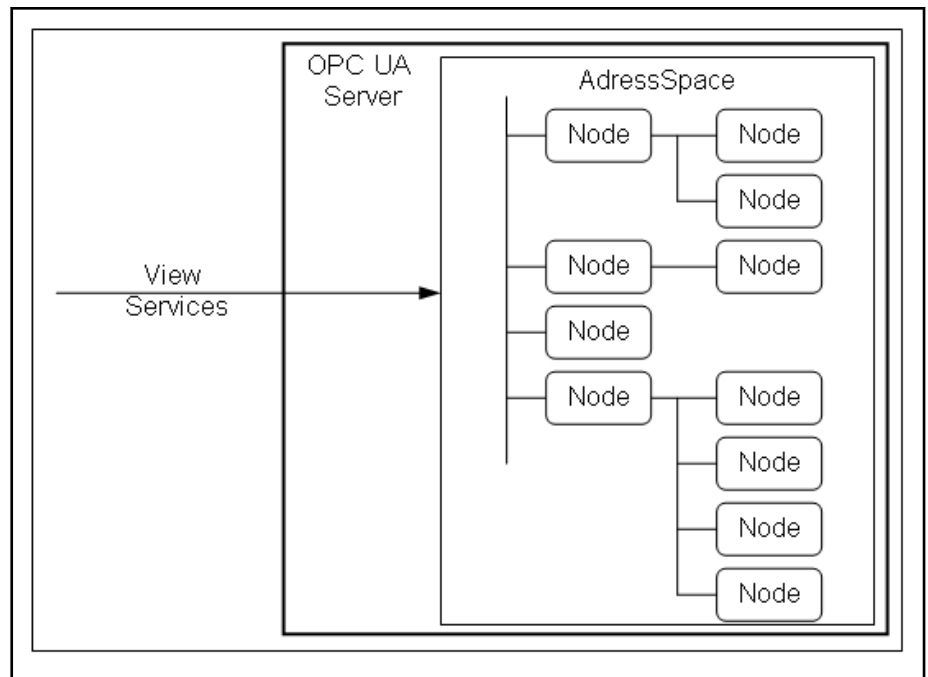


Fig. 4-117: View Service Set

Attribute Service Set

A node is composed of several attributes. This attribute serves to describe a node in detail. The node class specifies which attributes a node has. Only nodes of the Variable node class are assigned the attribute Value.

All node classes have a unique set of basic attributes. The basic attributes are: the NodeID to address a node, the NodeClass and the BrowseName.

Using the Service Set attribute, read and write access to these attributes is possible.

Overview

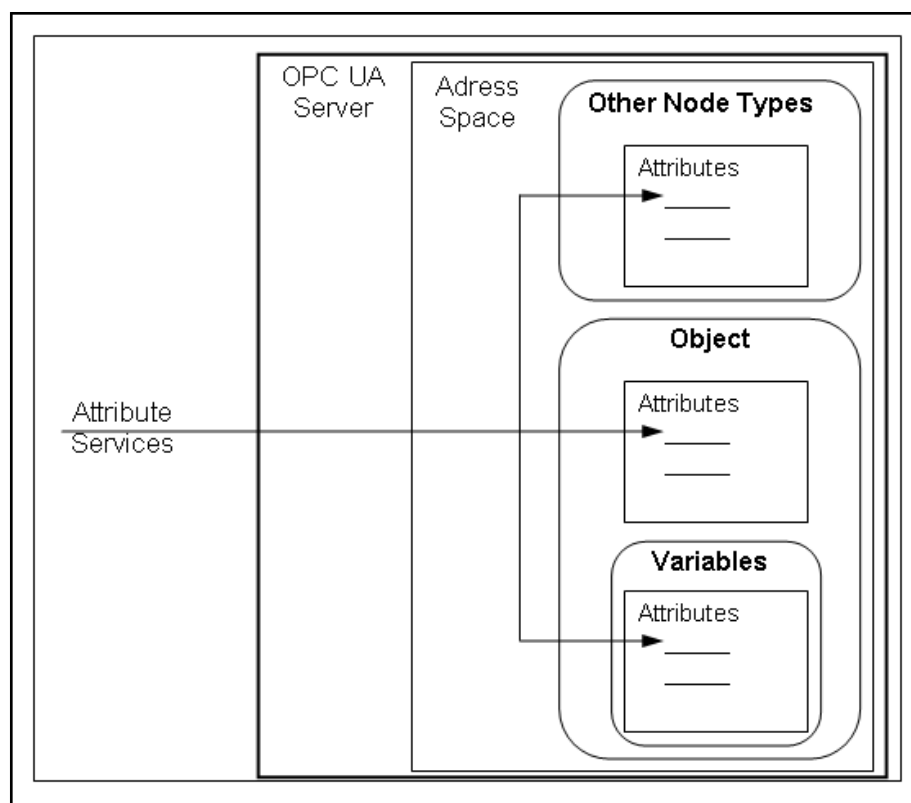


Fig.4-118: Attribute Service Set

Subscription Service Set

Subscriptions are an efficient method in OPC UA to receive information about value changes in the OPC UA server in the client. Using the Subscription Service Set, clients can create, modify and delete subscriptions.

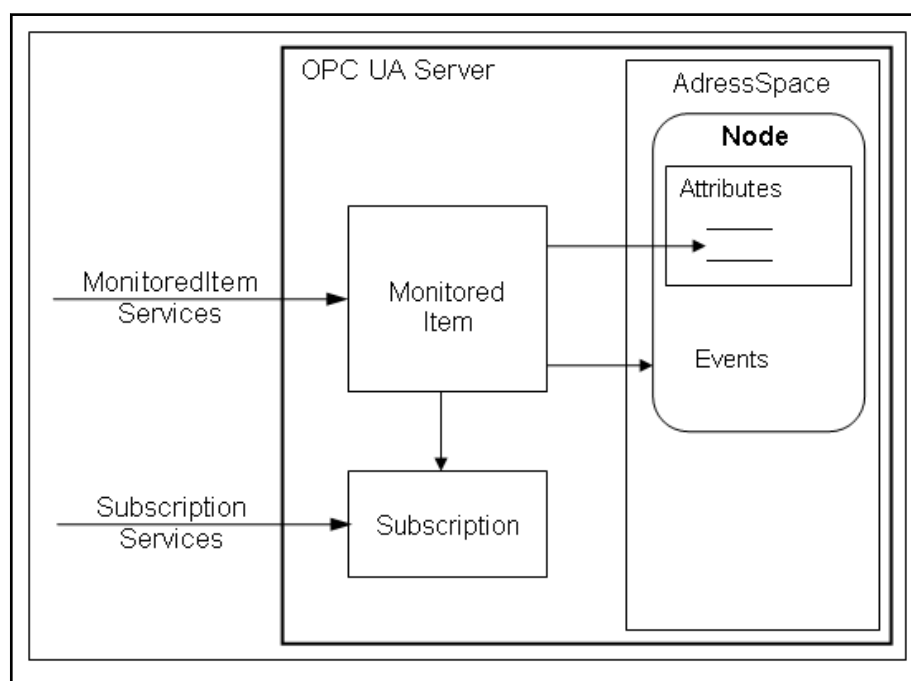


Fig.4-119: Subscription Service Set

MonitoredItem Service Set

MonitoredItems are node or event attributes that are part of a subscription. If the attribute value changes, it is automatically transmitted to the client.

Using the MonitoredItem Service Set, clients can create, modify or delete Monitored Items in the OPC UA server.

OPC UA Information Model

The OPC UA information model is based on nodes, describing an object-oriented context. A node can consist of attributes, methods and events. The content depends on the NodeClass. The type model allows mapping of all data types of user data up to meta data. Each node has a unique NodeID. The node connections are described by references. The reference semantics is specified by the ReferenceType.

The OPC UA information model is used as basis by other organizations such as PLCOpen and is extended as individual profile. More information about these extensions can be found under the index word Companion Specification or in the respective organizations or the OPC foundation. Each OPC UA server reports which profile it supports upon request.

Bosch Rexroth OPC UA Server

General Information

The Bosch Rexroth OPC UA server is a firmware-integrated component. It ensures a safe connection via a standardized communication protocol for access to the control. Adjustments or driver developments in the client are not required. The OPC UA server is initialized and started upon start-up of the control.

Bosch Rexroth OPC UA Server Properties

In the following, the Bosch Rexroth OPC UA server properties are described.

Properties

EndpointURL	The EndpointURL is part of the Endpoint. It is composed of the communication protocol and the host address. OPC UA defined a high-performance binary protocol (opc.tcp://) as a communication protocol and an XML-based protocol (http://). The Bosch Rexroth OPC UA server supports the binary protocol and used the fixed port 4840. EndpointURL structure: opc.tcp://<hostadresse>:4840
MessageSecurityMode	The safety level is specified using the MessageSecurityMode. <i>Possible values are:</i> • Sign: This safety level ensures the message integrity. The message can be protected against third party changes by adding a signature. • SignAndEncrypt: The message confidentiality is ensured in addition to the Sign safety level. Messages are encrypted using crypto algorithms. Thus, third parties do not have read access. • None: The messages are neither signed nor encrypted. The Bosch Rexroth OPC UA server supports the MessageSecurityMode None. Entries under SecurityPolicy are ignored.
SecurityPolicy	The SecurityPolicy describes the message encryption type. OPC UA defines the policies Basic128Rsa15 and Basic256. The SecurityPolicy is not supported.

Overview

PublishingInterval	The PublishingInterval defines the cycle in which a subscription tries to send a notification to a client. The PublishingInterval value is specified upon the subscription creation between the client and the server. The Bosch Rexroth OPC UA server uses the fixed PublishingInterval: 10ms, 100ms, 200ms, 500ms, 1.0s, 2.0s, 5.0s.
SamplingInterval	The SamplingInterval defines the cycle in which the server samples the subordinate data sources. The SamplingInterval value is specified between client and server upon creation of a MonitoredItem. The Bosch Rexroth OPC UA server uses the fixed PublishingInterval: 10ms, 100ms, 200ms, 500ms, 1.0s, 2.0s, 5.0s.

Fig.4-120: Bosch Rexroth OPC UA Server Properties



The PublishingInterval and the SamplingInterval describe values specified by the client in which the OPC UA server becomes active in the control. Time intervals that are too short can effect the overall behavior of the control!

PLC Namespace

The OPC UA server supports the client access to PLC data points. PLC variables can be read and changed via read and write services. The Publish service provides an efficient mechanism to inform a client via value changes of PLC variables. The PLC namespace can be analyzed using the browser service. Please note that the PLC namespace is based on the symbol configuration of the PLC program. Only symbolically exported data points are mapped in the OPC UA server name space (see symbol configuration of the IndraLogic documentation).

Central entry point to the PLC namespace is the Logic node

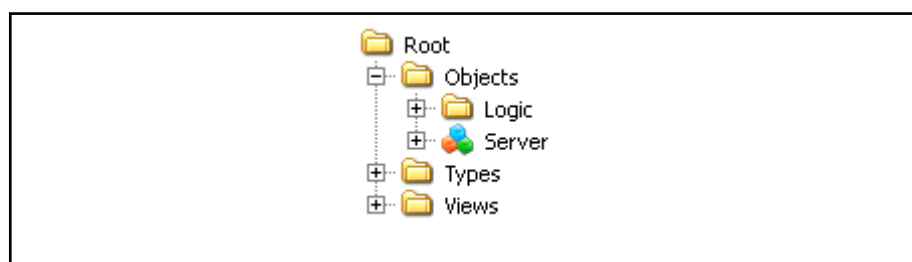


Fig.4-121: Root node logic

The structure under this node complies with the specification "PLCOpen: OPC UA For IEC 61131-3 1.00 Companion Specification" (can be obtained under <http://www.opcfoundation.org>). It corresponds to the PLC application structure.

OPC UA and the PLCOpen use a different basic data system. The types are mapped according to the following table.

Type Mapping

IEC 61131-3	OPC UA "build in" data type	Extended OPC UA data types by "OPC UA For IEC 61131-3 1.00 Companion Specification"	Comment
BOOL	Boolean		8 bit data type
SINT	SByte		8 bit data type
USINT	Byte		8 bit data type
INT	Int16		16 bit data type, signed

IEC 61131-3	OPC UA "build in" data type	Extended OPC UA data types by "OPC UA For IEC 61131-3 1.00 Companion Specification"	Comment
UINT	UInt16		16 bit data type, unsigned
DINT	Int32		32 bit data type, signed
UDINT	UInt32		32 bit data type, unsigned
LINT	Int64		64 bit data type, signed
ULINT	UInt64		64 bit data type, unsigned
BYTE	Byte	BYTE	BYTE is a data type specified by PLCOpen and derived by the byte with a bit string of length 8.
WORD	UInt16	WORD	WORD is a data type specified by PLCOpen and derived by the UInt16 data type with a bit string of length 16.
DWORD	UInt32	DWORD	DWORD is a data type specified by PLCOpen and derived by the UInt32 data type with a bit string of length 32.
LWORD	UInt64	LWORD	LWORD is a data type specified by PLCOpen and derived by the UInt64 data type with a bit string of length 8.
REAL	Float	-	32 bit floating point
LREAL	Double	-	64 bit floating point
STRING	String	STRING	STRING is a data type specified by PLCOpen and derived by the string. It is a one byte character string of variable length.
WSTRING	String	-	A UTF 8 string of variable length
DT DATE_AND_TIME	DateTime	-	64 bit data type representing the time in 100ns intervals since 1. January 1601.
DATE	DateTime	DATE	DATE is a data type specified by PLCOpen and derived from the DateTime 64 bit. It represents a date
TOD TIME_OF_DAY	DateTime	TOD	TOD is a data type specified by PLCOpen and derived from the DateTime 64 bit. It represents a time.
TIME	Double	Duration	Duration is a data type specified by PLCOpen and derived from the double 64 bit. It represents a time interval in milliseconds.

Fig.4-122: Type mapping IEC 61131-3 OPC UA

OPC UA Communication Sample

Introduction

The following chapter shows how to access the Bosch Rexroth OPC UA server with simple means. It is described how to parameterize a connection and how the different services of the server can be used.

Overview

Requirements

To simulate the communication sample, UaExpert of the Unified Automation GmbH is required. UaExpert is implemented for the Win32 and Linux platforms and can be downloaded as freeware via <http://www.unified-automation.com>.

This documentation refers to version 1.2.0 132 of UaExpert.

UaExpert

UaExpert is a universal OPC UA test client. By means of this test client, the OPC UA server address space can be analyzed and simple tests can be conducted with the OPC UA server.

The UaExpert supports

- browsing of address spaces,
- reading and writing of nodes (value attribute),
- monitoring of value changes and events (subscription),
- calling methods,
- reading of historic data

The UaExpert is divided into several windows.

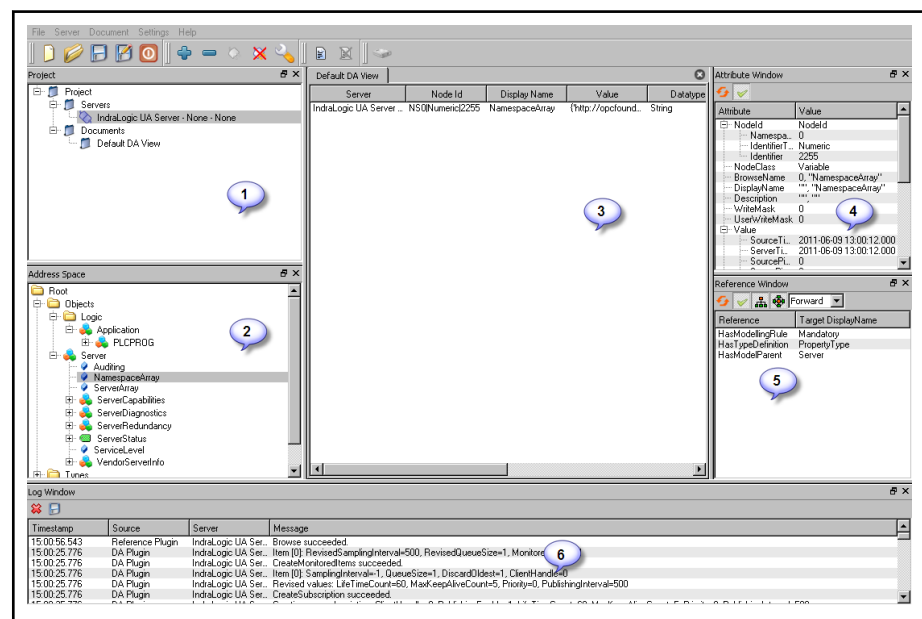


Fig.4-123: UaExpert

The subwindows have the following meaning:

- ① Project Window. Server connections and documents are managed here.
- ② Address Space Window. By means of the Address Space Window, navigating and browsing in the address space of the server selected in the Project Window is possible.
- ③ Data View Window. The Data View Window is used to map server data.

The following views are supported:

- DataAccess View

The DataAccess View describes the content of a subscription. All nodes monitored for a value change are listed.

- Alarms&Condition View
In the Alarms&Condition view, events and alarms triggered by the server are configured and displayed. Alarms and events are not supported in the current Bosch Rexroth OPC UA server version.
- HistoricalTrend View
In the HistoricalTrend view, the chronological value change can be displayed or parameterized for one or several nodes as diagram.
- Performance View
By means of the Performance View, an OPC UA server performance can be analyzed. The Read, Write, CreateMonitoredItem and DeleteMonitoredItem services are supported.
- ④ Attribute Window
In the Attribute View, all attributes of the nodes selected in the Address Space Window.
- ⑤ Reference Window
The Reference Window shows all references for the nodes selected in the Address Space Window. It is differentiated between forward references, references linking way from this node and inverse references, linking to this node.
- ⑥ Log Window
The Log Window is used to diagnose communication problems.

Establishing a Connection

An end point has to be specified to establish a connection to the OPC UA server. The end point consists of the communication protocol, the host address and the security settings.

To define an end point for an OPC UA server in UaExpert, select **Server ► Add...** In the dialog, enable the "Discovery" tab und select "<Double click to Add Server...>" in the tree view under "Custom Discovery".

Overview

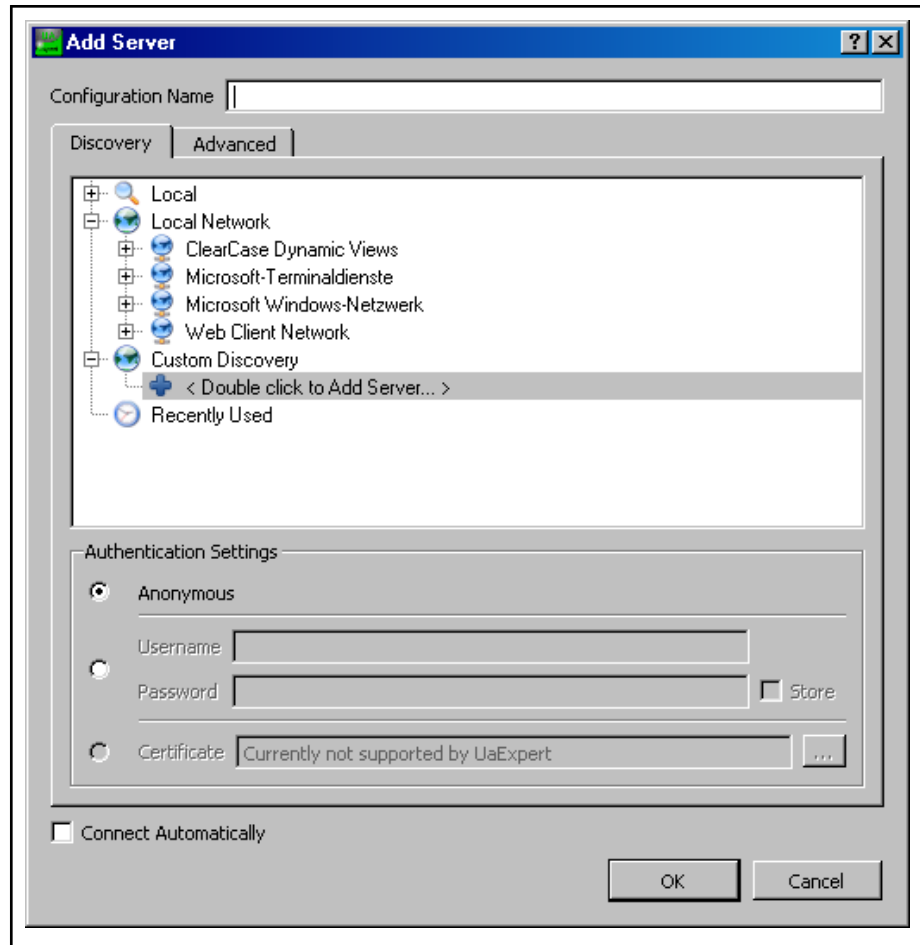


Fig.4-124: Dialog: Add Server

By double-clicking on the entry, the entry mask of the communication protocol and the host address is displayed. The communication protocol is already assigned as UA binary protocol by the entry "opc.tcp://" in the input line. Add the IP address or the host name for the OPC UA server. Optionally, the port number, separated by a colon can also be specified. The Bosch Rexroth OPC UA server uses port 4840.

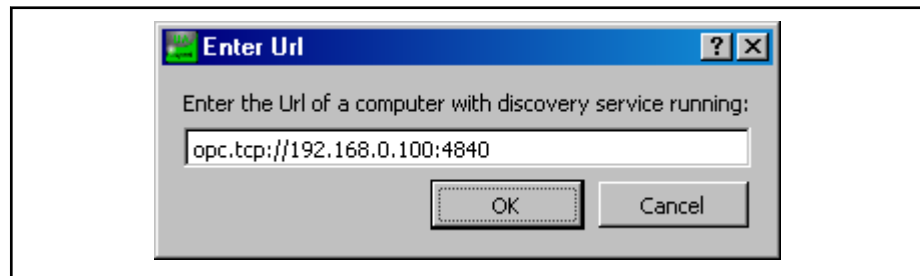


Fig.4-125: Entering the EndpointUrl

By pressing on **Ok**, you can return to the previous dialog.

The EndpointUrl is now displayed under the "Custom Discovery" entry.

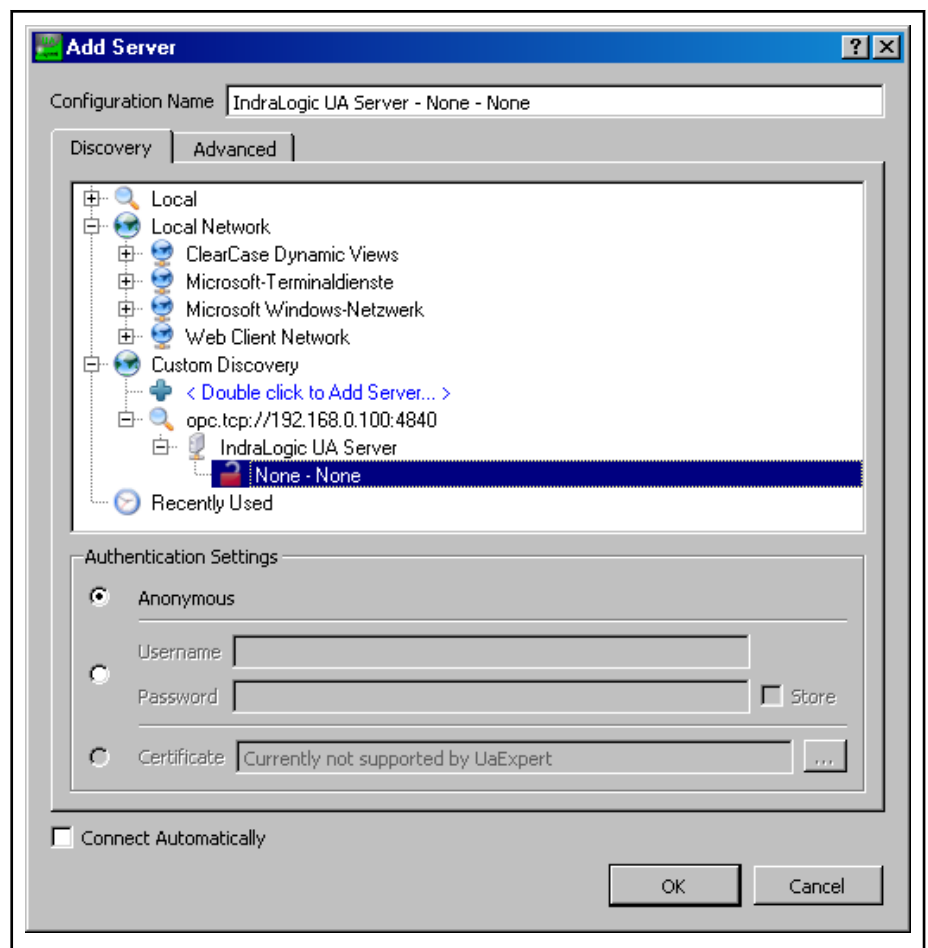


Fig. 4-126: Selection "No Security Endpoint"

As soon as the communication to the server has been established, the entry can be expanded as tree. The ServerUri "IndraLogic UA Server" and the security end points should be visible. The Bosch Rexroth OPC UA server does not support any security. Only the entry "None - None" is available. Click on the entry and close the dialog.

Browsing

The ServerUri is now visible for the parameterized connection in the Project Window below the "Server" node. A connection to this OPC UA server can now be established via the context menu of this entry or via the "Server Connect" menu item.

After the connection has been established successfully, the address space of the OPC UA server is displayed in the Address Space Window.

Overview

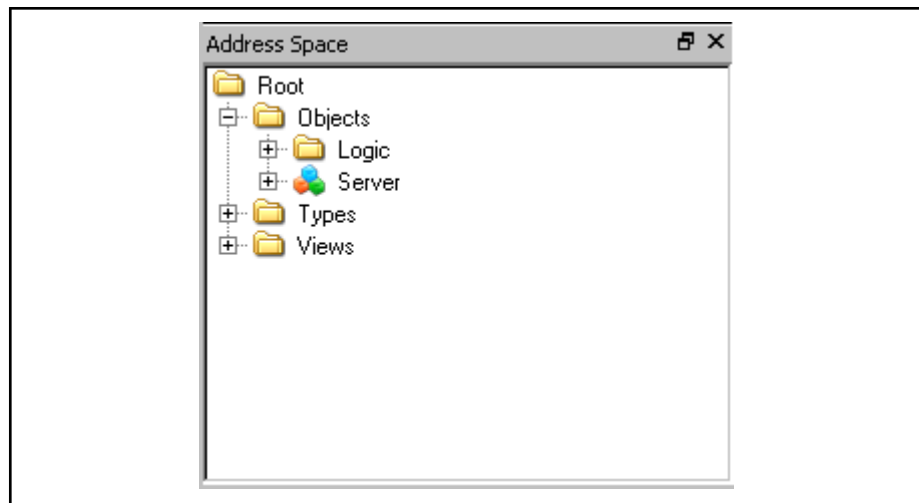


Fig.4-127: Address Space Window

The objects in the tree view are OPC UA nodes. These nodes can now be expanded and facilitate navigation in the address space of the OPC UA server. All nodes contained in the address space are displayed.

By selecting a node via mouse-click, all node attributes are listed in the Attribute Window including the values. The values in this window are read-only. This means that a client cannot change these values.

All references from the node (forward) and all references to this node (Inverse) can be read in the Reference Window.

Subscribe and Write

The UaExpert supports the mechanism to monitor changes in data (subscription) by the DataAccess View in the Data View Window. One or several views can be generated via the menu **Document - Add** and by selecting "Data Access Classic View" in the "Add Document" dialog.

Nodes and in particular node attributes that are monitored for value changes are called Monitored Item. To generate a Monitored Item, the node in the Address Space Window has to be dragged to the Data View Window using Drag and Drop.

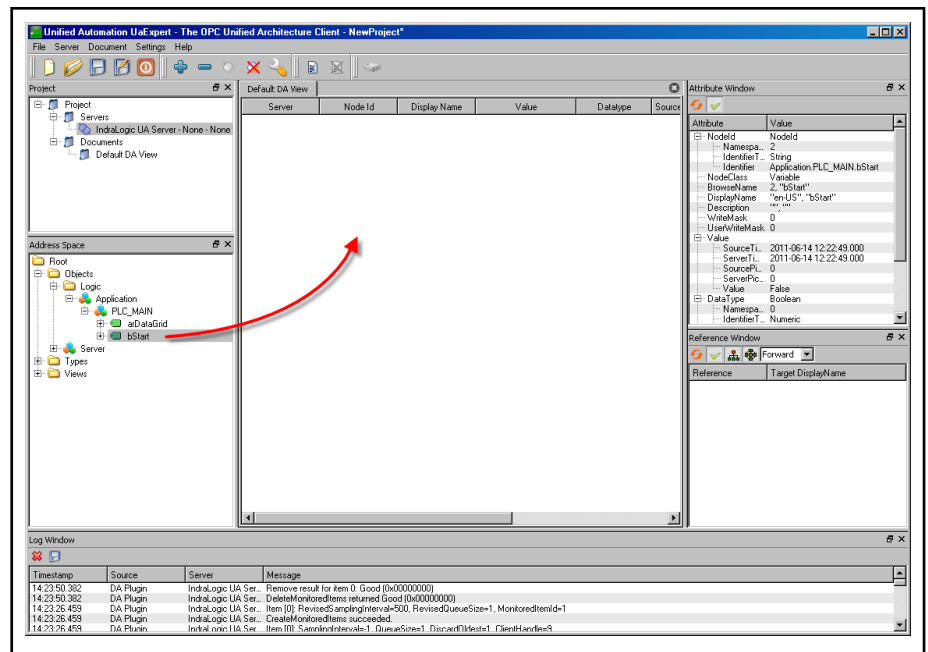


Fig.4-128: Create Monitored Item

The attribute Value is automatically used as Monitored Item by UaExpert. Thus, the nodes have to be part of the node class Variable.

The node value is now displayed in the Data View Window and is regularly updated. A "Data Access Classic View" can be viewed as a Subscription. The dialog to set the subscription properties can be accessed via the context menu of this window. Right-click on the Data View Window and select "Subscription Settings..."

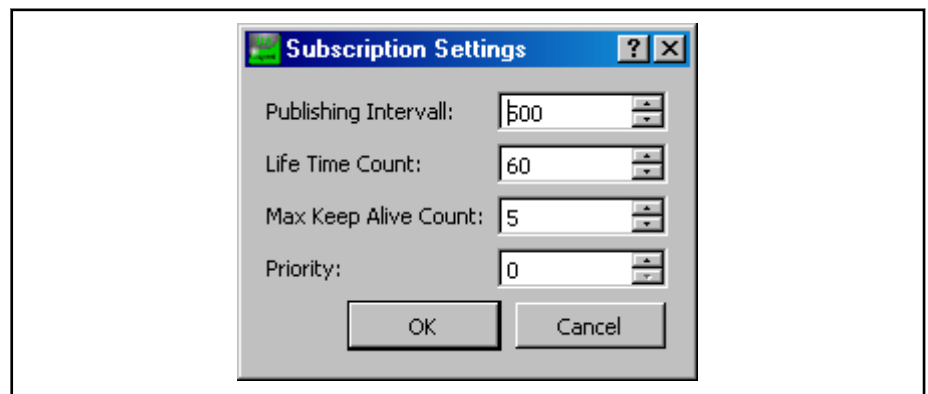


Fig.4-129: Dialog: Subscription Settings

The following settings can be selected in the dialog. Please note that all values are negotiated between client and server.

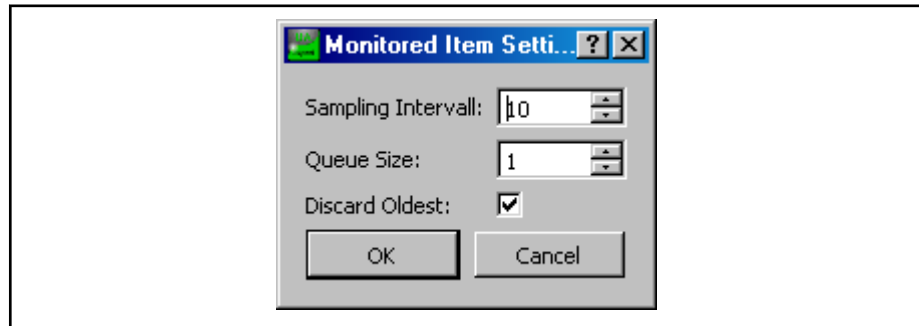
Overview

Setting values for a subscription

Publishing Interval	The Publishing Interval describes the interval in milliseconds in which value changes are transmitted to the client. The Bosch Rexroth OPC UA server uses fixed Publishing Intervals: 10ms, 100ms, 200ms, 500ms, 1.0s, 2.0s, 5.0s.
Life Time Count	The Life Time Count serves to monitor the client/server connection. It specifies how often the Publishing Interval expires without a connection being established between client and server. If the end of the Life Time Count is reached, all Monitored Items of this subscription and the subscription itself are deleted from the server.
Max Keep Alive Count	The Max Keep Alive Count specifies how often the Publishing Interval can expire without a value of the Monitored Items of the subscription changing. The client is thus not informed about the value changes. If the Max Keep Alive Count is reached, the client is informed in a "blank" message that the subscription is still active.
Priority	A priority can be assigned to a subscription via this value. If several subscriptions are active in a server, subscriptions with a higher priority are also processed with priority.

Fig.4-130: Subscription settings

Further setting options refer to the Monitored Items. The settings can be selected by right-clicking on the Monitored Item and selecting the "Monitored Items Settings...". The values are also negotiated between client and server.

*Fig.4-131: Dialog: Monitored Item Settings*

Setting values for Monitored Items

Sampling Interval	The Sampling Interval specifies in which interval the server samples the subordinate data source to determine a value for the Monitored Item. If the value is set to -1, the Publishing Interval is used.
Queue Size	Using this entry, the Queue size can be set for the Monitored Item. All Queue values are transmitted in the Publishing Interval.
Discard Oldest	Discard Oldest describes the rule for the server behavior in case of a full Queue. If the value is set, the oldest value of the Queue is replaced by a new value. Otherwise, the new value is discarded.

Fig.4-132: Monitored Item settings

To write a value, double-click in the value column of a Monitored Item. The value can now be edited. After clicking on Enter, the new value is written to the server.

4.4.6 Modbus TCP Introduction and Overview

The Modbus TCP is used for the data exchange between EtherNet devices supporting the Modbus protocol. It can be used to connect several HMI devices to a control. The Modbus slave (server) is the control used for the Modbus masters (clients) to connect.

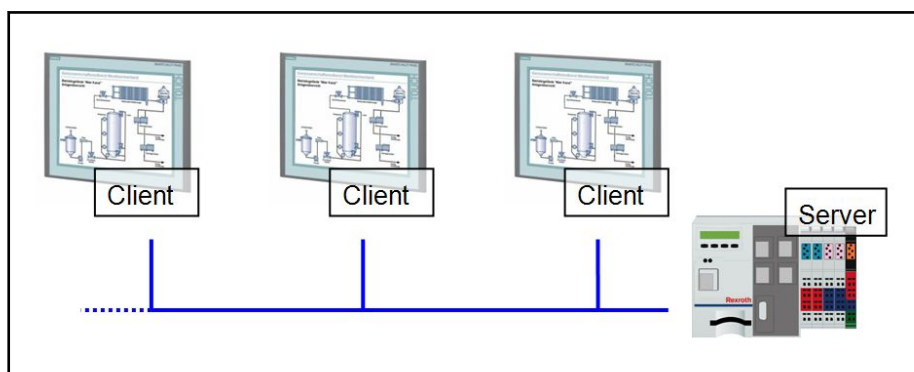


Fig.4-133: Connecting several HMI devices (Modbus clients) to a control (Modbus server)

Modbus TCP differentiates between four different data types:

- Discrete input: bit-addressed binary input (array of Boolean)
- Coil: bit-addressed binary output (array of Boolean)
- Input Register: Word-based output (array of word)
- Holding Register: Word-based output (array of word)

Transmitting medium	Ethernet
Protocol	Modbus TCP on TCP-Port 502
Max. number HMI devices	16 per Modbus TCP server

Overview

Transmission efficiency	1000 I/O operations per second for 1ms cycle time
I/O area	65536 Bit Discrete Input 65536 Bit Coil 65536 Word Input Register 65536 Word Holding Register

Fig.4-134: *Modbus TCP properties*
The IndraMotion MLC and IndraLogic XLC controls support the following Modbus TCP-compatible HMI devices:

- Siemens WinCC-flexible 2007
- Pro-face GP-Pro EX2.2.

Devices of other manufacturers can also be supported. However, this has not been tested.

A Modbus TCP server is implemented in each of the "IL_ModBusTCPServer" and "IL_ModBusTCPServerType02 " function blocks contained in the RIL_ModBusTCP.library.

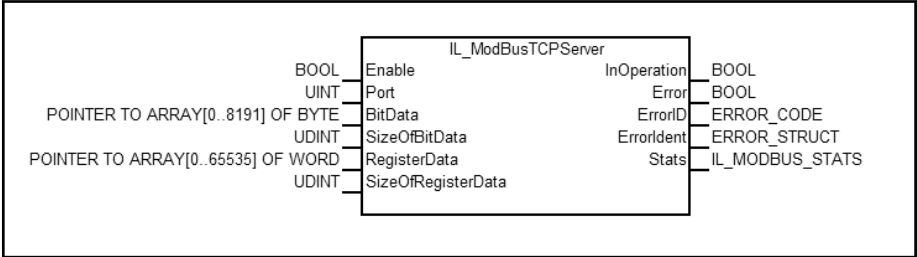


Fig.4-135: *IL_ModBusTCPServer function block*

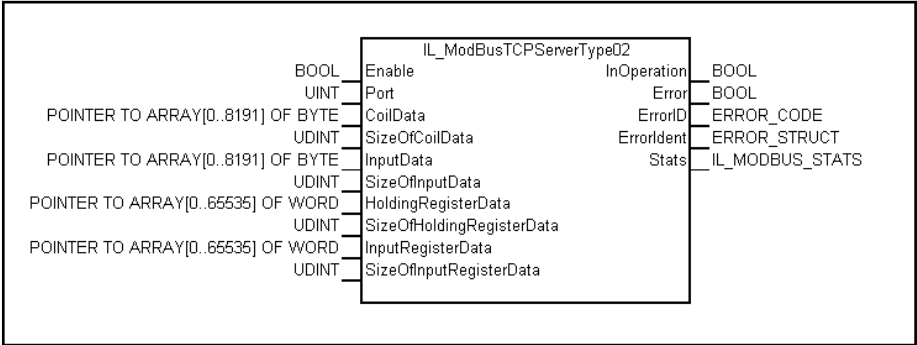


Fig.4-136: *IL_ModBusTCPServerType02 function block*

 The functions blocks IL_ModBusTCPServer and IL_ModBusTCPServerType02 differ in the following:

In contrast to the Modbus specification, the data ranges for "Coil Data" and "Discrete Input Data" are summarized under "BitData". The "Holding Register" and "Input Register" are summarized under "RegisterData" in the "IL_ModBusTCPServer" function block.

A more detailed description of the Modbus TCP functionality can be found in the documentation

- Rexroth IndraWorks xxVRS, Basic Libraries IndraLogic 2G
- xx = Version

5 Application Samples

5.1 Overview

This chapter describes simple sample applications for individual visualization types. The short character of the samples is intentional. For more detailed information, please refer to the corresponding documentations of the visualization systems.

5.2 Symbol Configuration

5.2.1 General

In order for an OPC client to access the PLC variables of a control, the variables required by the OPC client or the variables provided for access by the PLC project have to be configured in the symbol configuration of IndraLogic.

The variables can be configured using the following access authorizations:

- No access
- Read
- Write
- Read and write

Symbol configuration can be carried out with a wizard via the "Symbol Configuration" window, via "pragma directives" in the variable declaration section of the POU or via variables lists.

A prerequisite for symbol configuration is that the PLC object "Symbol configuration" exists below the PLC application in the IndraWorks project tree. The object listed in the following table from the "PLC Objects / General Objects" library window has to be added below the "Logic / Application / HMI" device node using drag & drop. The following figure shows the Project Explorer once the object has been added.

Object type	Name	Description
Symbol configuration	"_"	Configuration of the variables for access via the OPC server with access rights.

Fig. 5-1: Adding PLC library object "Symbol configuration"

Application Samples

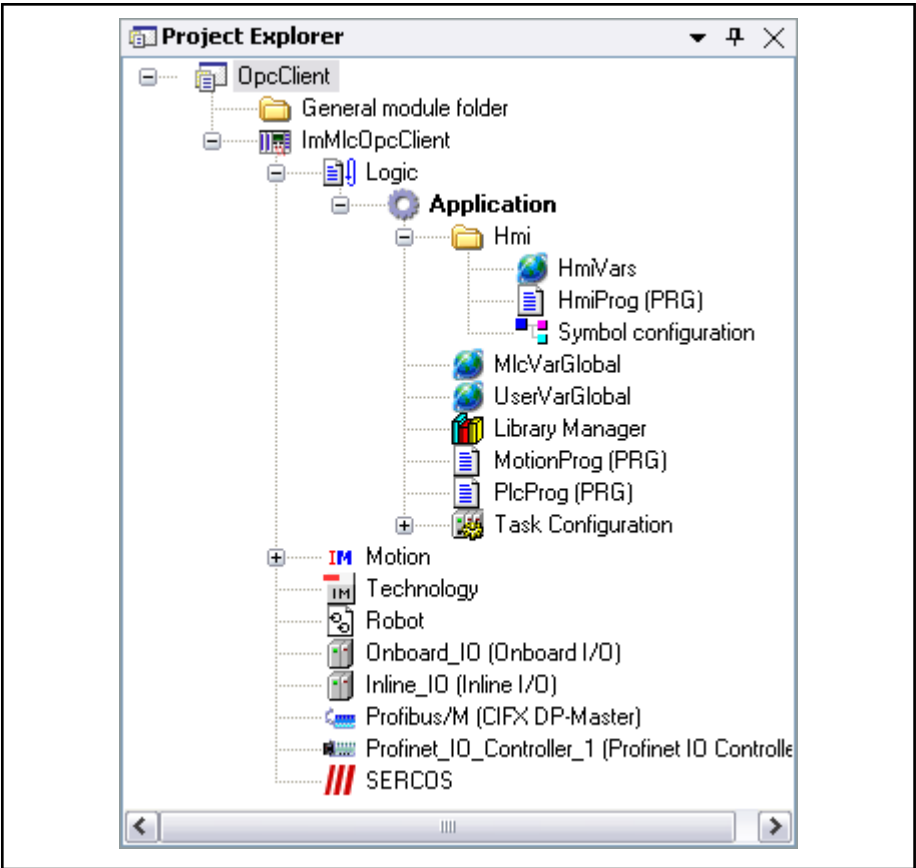


Fig.5-2: Added library object "Symbol configuration"

5.2.2 Symbol Configuration Using a Wizard

The symbol configuration wizard is opened via the newly created "Symbol Configuration" node in the IndraWorks Project Explorer.

The symbol configuration window is divided into two areas:

Available variables	List of all of variables in the PLC program with the respective maximum access authorization level. If no variables are listed, use the "Refresh" link to update the list. Note that it must be possible to compile the project without errors.
Selected variables	List of all variables which were configured for access via the OPC server, with the respective configured access authorization. Select the "Display invalid variables" link to highlight all variables that no longer exist in the PLC project.

Fig.5-3: Structure of the "symbol configuration wizard"

Select or delete variables by double-clicking on the element or by using the ">" and "<" buttons. This selects or removes the selected element and all subordinate elements.

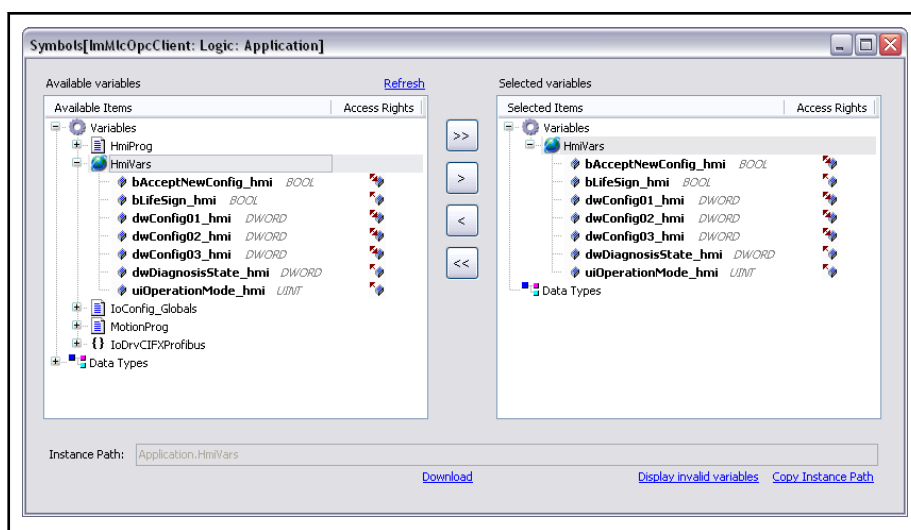


Fig. 5-4: Symbol configuration wizard



The "Symbol configuration" window lists only those variables which are used in the PLC program (POU), if this is called in a task. This means that variables which are not used are not in the project tree.

The access rights for the respective variable can be changed by clicking on the symbol in the column "Access rights" in the section "Selected variables". Access rights are switched in the following order:



Read and write access



Write Access



Read access

5.2.3 Symbol Configuration Using a Pragma

Symbol configuration using "pragma directives" is carried out directly in the declaration area of the variables, e.g. in variable lists or in POU's.

The "{attribute 'symbol'}" attribute with the parameters for access authorization is available for pragma configuration. The possible configurations are shown in the following figure.

Pragma directive	Description
{attribute 'symbol' := 'none'}	Variables are blocked for access and are not displayed in the OPC server browsing list.
{attribute 'symbol' := 'read'}	Variables are only enabled for read access and are displayed in the OPC server browsing list.

Application Samples

Pragma directive	Description
{attribute 'symbol' := 'write'}	Variables are only enabled for write access and are displayed in the OPC server browsing list.
{attribute 'symbol'} {attribute 'symbol' := 'readwrite'}	Variables are enabled for read and write access and are displayed in the OPC server browsing list.

Fig. 5-5: Pragma directives for "symbol configuration"

Depending on the position or the line in which the Pragma is written, it is valid for all variables in the declaration range or for just one variable.

- In order for the pragma to affect only a single variable it has to be written into the line before the variable.
- If it is to affect all variables in the declaration range, it has to be in the first line of the declaration. In this case individual variables can still be assigned a different access authorization.

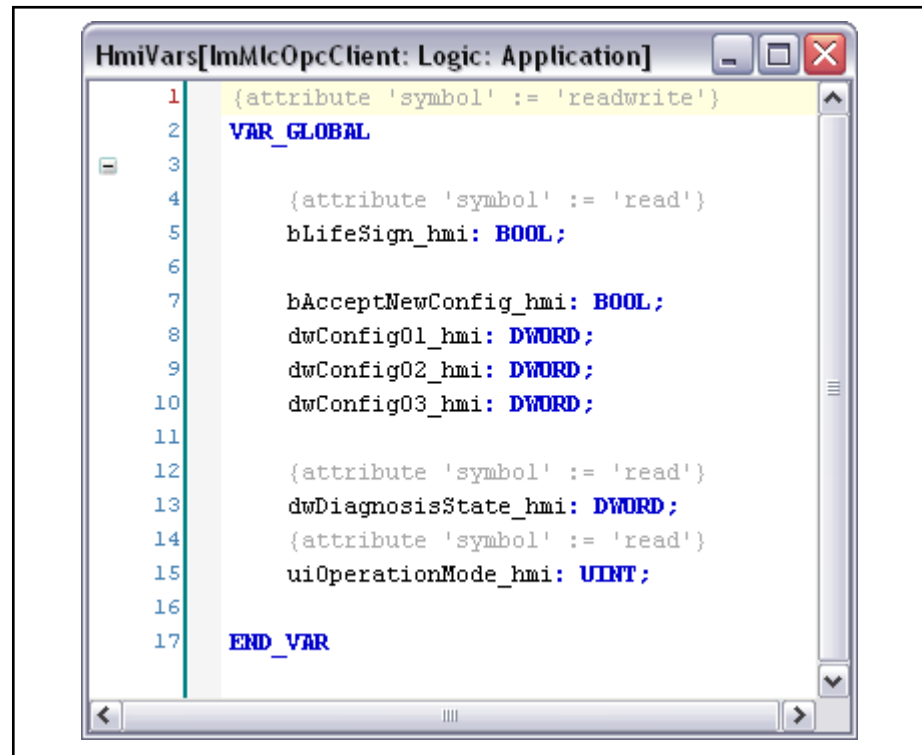


Fig. 5-6: HMI variable declaration with pragma



If symbol configuration is carried out via pragma directives these symbols are **not** displayed in the symbol configuration dialog.

When the PLC application is downloaded, the symbol information is created in the IndraWorks "IndraLogic" project subdirectory as an XML file and is loaded into the control as a symbol application.

5.3 VI Composer

In preparation

5.4 WinStudio with IndraLogic XLC

5.4.1 General

The aim is to create a WinStudio interface that can be used to control a drive and display the control status information.

This project is a first step to using "WinStudio" in connection with IndraLogic XLC or IndraMotion MLC. The "WinStudio" interface is created step by step.

Subtasks

Task description:

- Creating an IndraWorks project in which a drive is operated,
- Integrating WinStudio in IndraWorks,
- Creating the WinStudio application to visualize data of the control project (operating panel on the control).
- Start the interface.

5.4.2 Technical Requirements

Requirements

- PC with installed IndraWorks from version 11,
- IndraLogic XLC or IndraMotion MLC as control with sercos interface,
- IndraDrive C (MPx-07VRS) as drive,
- WinStudio V7.2-RCS under IndraWorks,
- IndraWorks Engineering 12Vxx.

Meeting these prerequisites is described in

- Rexroth IndraLogic XLC 12VRS Functional Description
DOK-IM*ML*-SYSTEM**V12-RExx-EN-P and
- IndraWorks Online Help, **IndraLogic XLC/IndraMotion MLC system ▶ XLC/MLC Functional Description ▶ Commissioning and Operation**,

HMI, type VEP xx.3 or VPP xx.3

- The current firmware is loaded.
- The IP address is known,
- the control and the HMI are connected to the programming device (PC) via Ethernet.

Prerequisites:

- A bridged sercos bus if there is no drive. Use a virtual axis instead of a real axis in this case.

5.4.3 Creating the IndraWorks Project

Start the IndraWorks Engineering environment using the icon on the desktop or the respective selection in the Windows program folder.

To create a new IndraWorks project, select

File ▶ New ▶ Project....

The dialog "Create New IndraWorks Project" appears".

Enter the name "WinStudio" (any name accepted by IndraWorks can be chosen) for the project and select a directory to store the project in.

Application Samples

Insert the "IndraMotion XLC Lxx" control into the project by dragging the device from the library to the project. Please note that the device can only be dragged directly into the project folder.

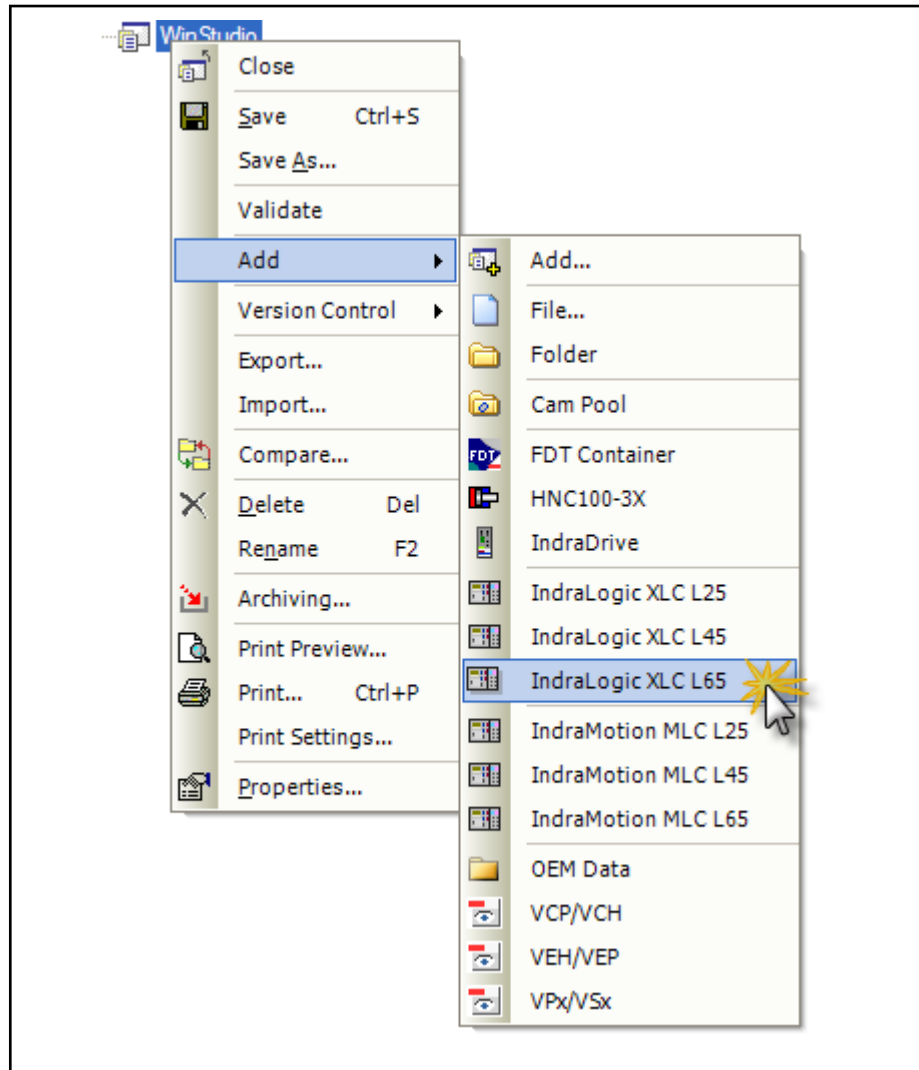


Fig.5-7: Selecting the IndraLogic XLC L65 control

Specify your device name (IndraWorks will notify you if any characters chosen are invalid) or use the one already specified.

Click on "Next" to make the communication settings and hence, establish a connection to the control.

The settings under "Firmware version" and "Firmware release" have to be adapted to the control.

The option "CML65.1-3P" has to be selected under device type.

Enter the IP address of your control into the field **IP address**. The IP address of the control can be found on the IndraLogic XLC VPP under "Shortcut to RxSettings".

If the option "Safe Online Mode" is enabled, another prompt is displayed before any safety-relevant activities can be carried out.

Subsequently, a **connection test** to the control should be carried out.

If the connection test was successful, determine which drives can be operated at your control.

Open the context menu of the "sercos" folder, **Scan bus configuration**.

"Drive1" was detected on the figure below.

By double-clicking on the symbol in column IPO (1), editing of the motion functionality in the control was selected.



Interpolation in the control  or drive  :

- Switching by double-clicking on the icon.
- The IndraDrive (interpolation in the drive) becomes a SercoDrive (interpolation in the control).

Its name was changed to "RA1" (2).

"Create Drives" (3) creates the drive in the Project Explorer.

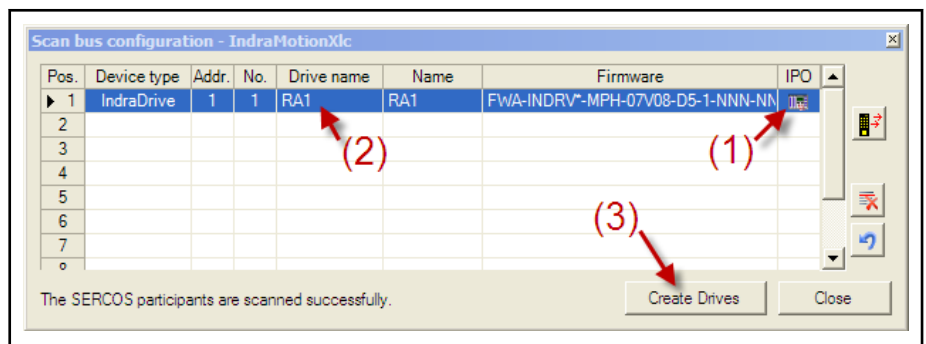


Fig.5-8: Drive scanning result

If control and drive have been inserted into the project, you should see the following in IndraWorks.

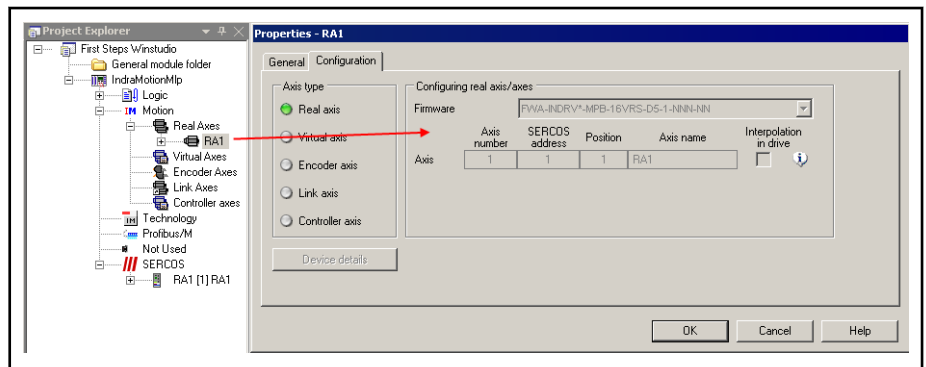


Fig.5-9: IndraWorks – Control and drive configured correctly

Create a control program in structured text under **Logic ► Application ► MotionProg (PRG)** . A double-click on the "MotionProg" file automatically opens IndraLogic in the IndraWorks middle part.

Application Samples

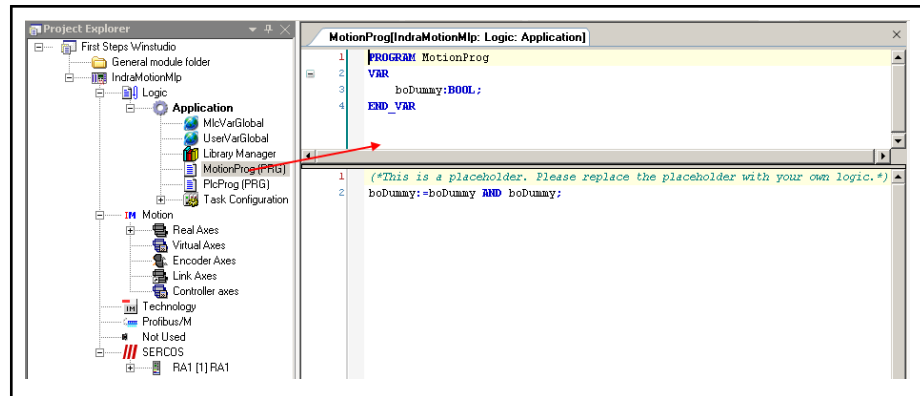


Fig.5-10: IndraLogic – MotionProg

Delete the code predefined in the file from the editor.

Insert the following function blocks into the program: MC_Power, MC_MoveVelocity, MC_Stop and MC_Reset.

Start with MC_Power. Use the F2 input assistance.

Enter the instance name (PowerRA1) in the declaration part and select type (User-defined Types ► ML_PLCopen ► POU ► Motion ► MC_Power) in the F2 input assistance.

Disable the option "Insert with Arguments".

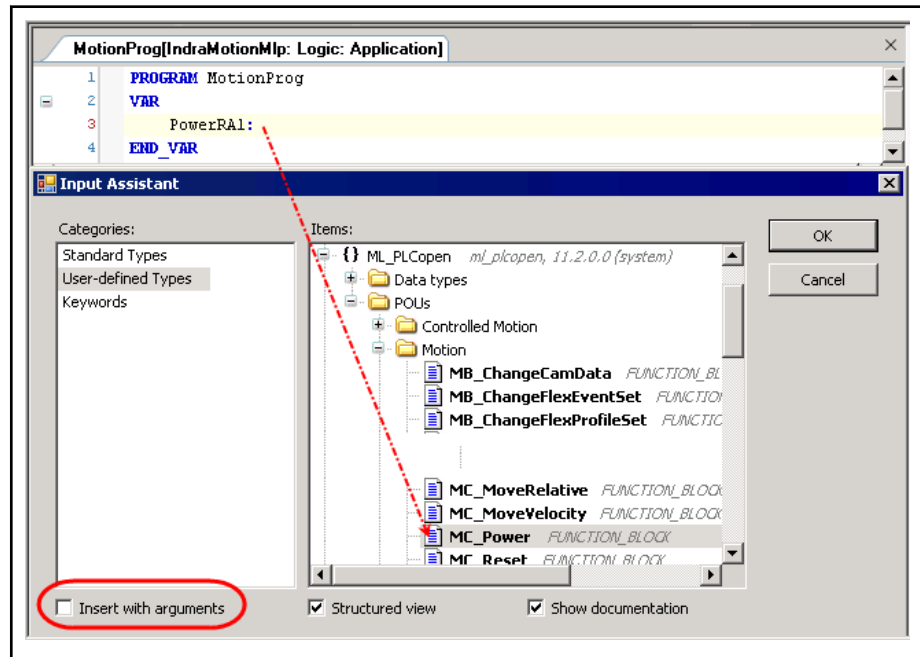


Fig.5-11: F2 input assistance, declaration, selection of the function block type (without argument)

Add ";" at the end of the declaration line.

Go to the implementation part and open the F2 input assistance (Instance Calls ► PowerRA1).

Enable the option "Insert with Arguments".

Application Samples

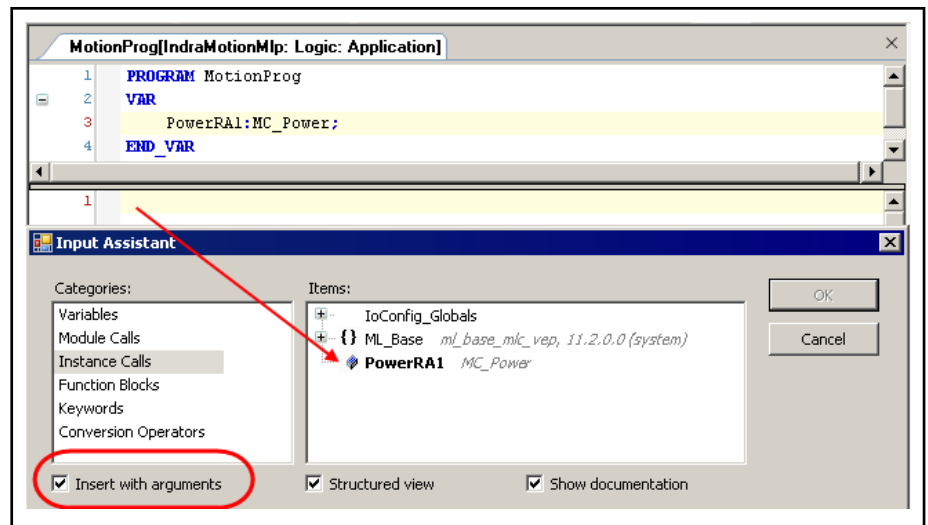


Fig.5-12: F2 input assistance, declaration, selection of the function block type (without argument)

We will provide the template of the block for wiring.

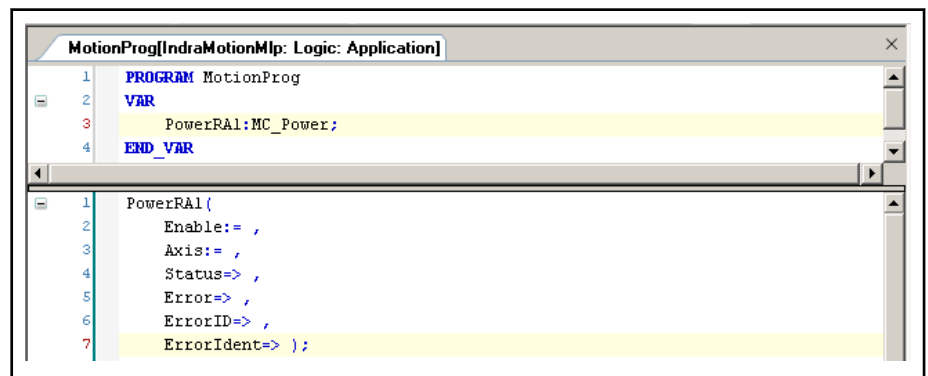


Fig.5-13: Implementation, template of the PowerRA1 block: MC_Power

Please assign "Enable" to "en_PowerRA1". When exiting the line, a declaration window is displayed in the Editor.

Please assign "Axis" to the corresponding axis reference. Use the F2 input assistance.

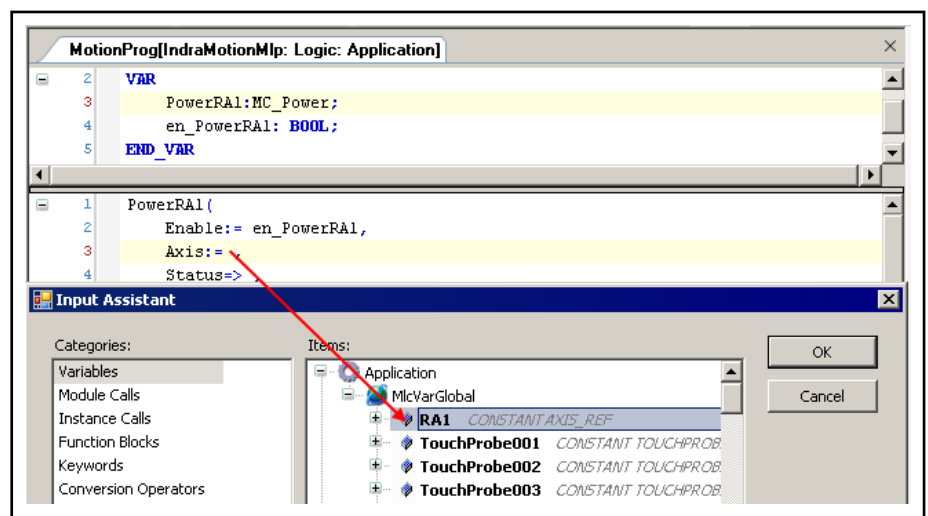


Fig.5-14: F2 input assistance, area of the global constants to be selected

Application Samples

The axis reference is entered as Global Variable (constant) under the name of the variable list and the axis description.

Complete all other variables.

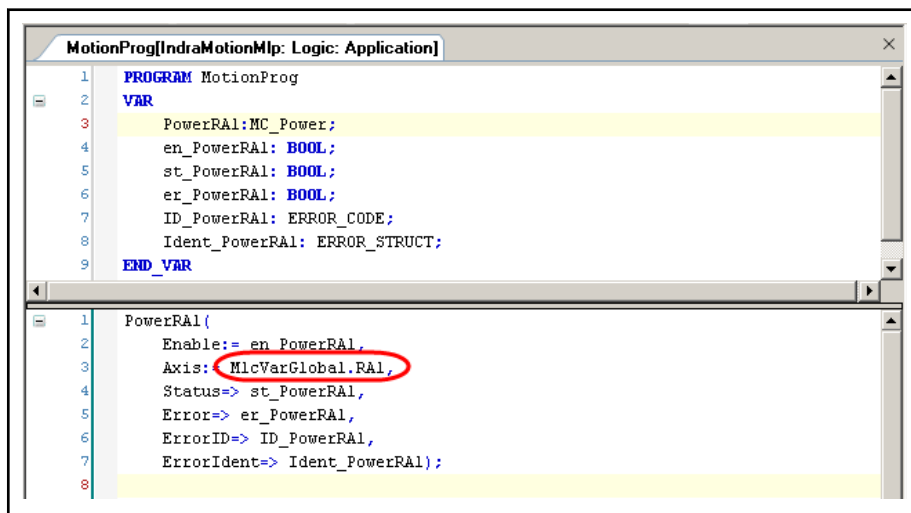


Fig. 5-15: PowerRA1: MC_Power: Complete in declaration and implementation.

Insert the following blocks into the program: MC_MoveVelocity, MC_Stop and MC_Reset.

```

1 // Power on, RA1
2 PowerRA1(
3     Enable:= en_PowerRA1,
4     Axis:= MlcVarGlobal.RA1,
5     Status=>st_PowerRA1 ,
6     Error=> er_PowerRA1,
7     ErrorID:=ID_PowerRA1 ,
8     ErrorIdent=> Ident_PowerRA1);
9 // Run RA1 with given velocity
10 MoveVelRA1(
11     Execute:= ex_MoveVelRA1,
12     Velocity:= 300,
13     Acceleration:= 20,
14     Deceleration:= 20,
15     Axis:= MlcVarGlobal.RA1,
16     InVelocity=> in_MoveVelRA1,
17     Active=> ac_MoveVelRA1,
18     CommandAborted=> ca_MoveVelRA1,
19     Error=> er_MoveVelRA1,
20     ErrorID:= ID_MoveVelRA1,
21     ErrorIdent=> Ident_MoveVelRA1);
22 // Stop RA1 with given deceleration
23 StopRA1(
24     Execute:= ex_StopRA1,
25     Deceleration:= 20,
26     Axis:= MlcVarGlobal.RA1,
27     Done=> do_StopRA1,
28     Active=> ,
29     CommandAborted=> ,
30     Error=> er_StopRA1,
31     ErrorID:= ID_StopRA1,
32     ErrorIdent=> Ident_StopRA1);
33 // Clear error, RA1
34 ResetRA1(
35     Execute:= ex_ResetRA1,
36     Axis:= MlcVarGlobal.RA1,
37     Done=> do_ResetRA1,
38     Error=> er_ResetRA1,
39     ErrorID:= ID_ResetRA1,
40     ErrorIdent=> Ident_ResetRA1);
41

```

Fig. 5-16: MotionProg – Structured text

The part of the declaration containing the variables is filled during the program creation.

Application Samples

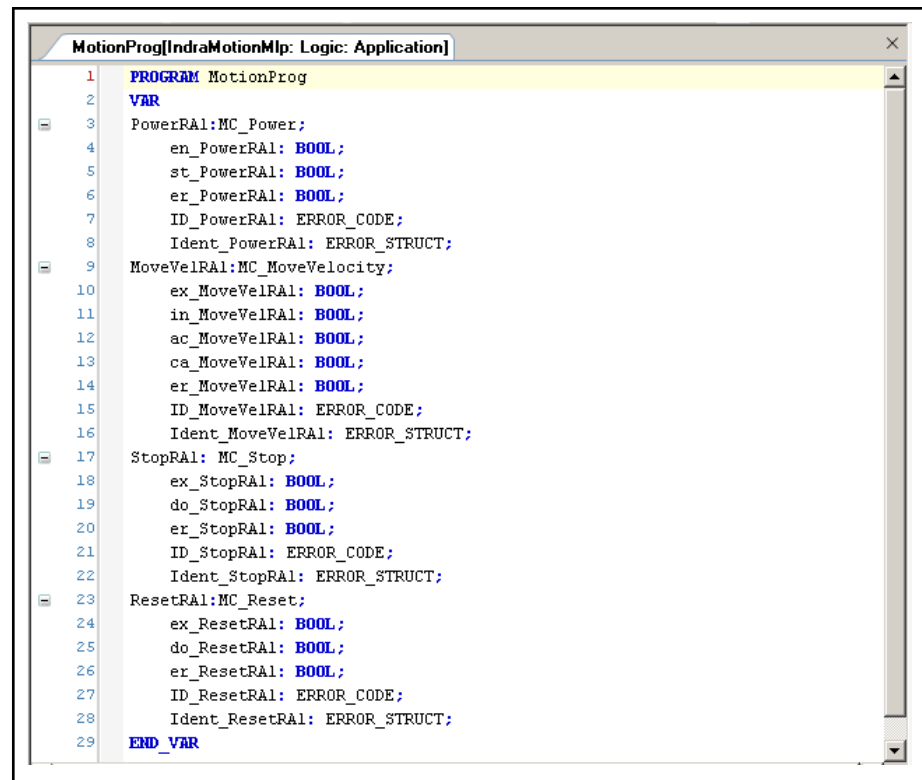


Fig.5-17: MotionProg – Variables

If the control program was created, it should be compiled:

1. **Create ► Compile Application** to ensure that there are no errors in the source code.
2. If the compilation has been executed successfully (see task list), go on-line with the control (**right mouse button on Control ► Switch online** or click on the icon in the toolbar).
3. A subsequent phase switching from P2 to BB is required.
In the context menu of the sercos node, select **SERCOS state ► P4**.
4. At the end, switch the application online (the program created in the IndraLogic uploaded to the PLC **Debug ► Login**) and start the program (**Debug ► Start**).

The control program creation is now completed. The variables enabled via the WinStudio interface at a later point in time can be seen in the IndraLogic.

5.4.4 Integrating the WinStudio Visualization in IndraWorks

To access program variables externally (as it is the case with WinStudio, external application), symbols with certain access rights must be defined. This is executed via the symbol configuration.

This symbol configuration has to be added to the application (**Application ► Right mouse button ► Add ► Symbol Configuration**).



You can only add in offline mode.

If the symbol configuration is inserted under "Application", the available variables are shown on the left after the symbol configuration has been selected and after clicking on refresh (1). The variables can be moved to the **"Selected variables"** area using the arrow keys (2). The variables selected can only be

activated in an external application. If the variables were selected from the **MotionProg**, the following figure appears:

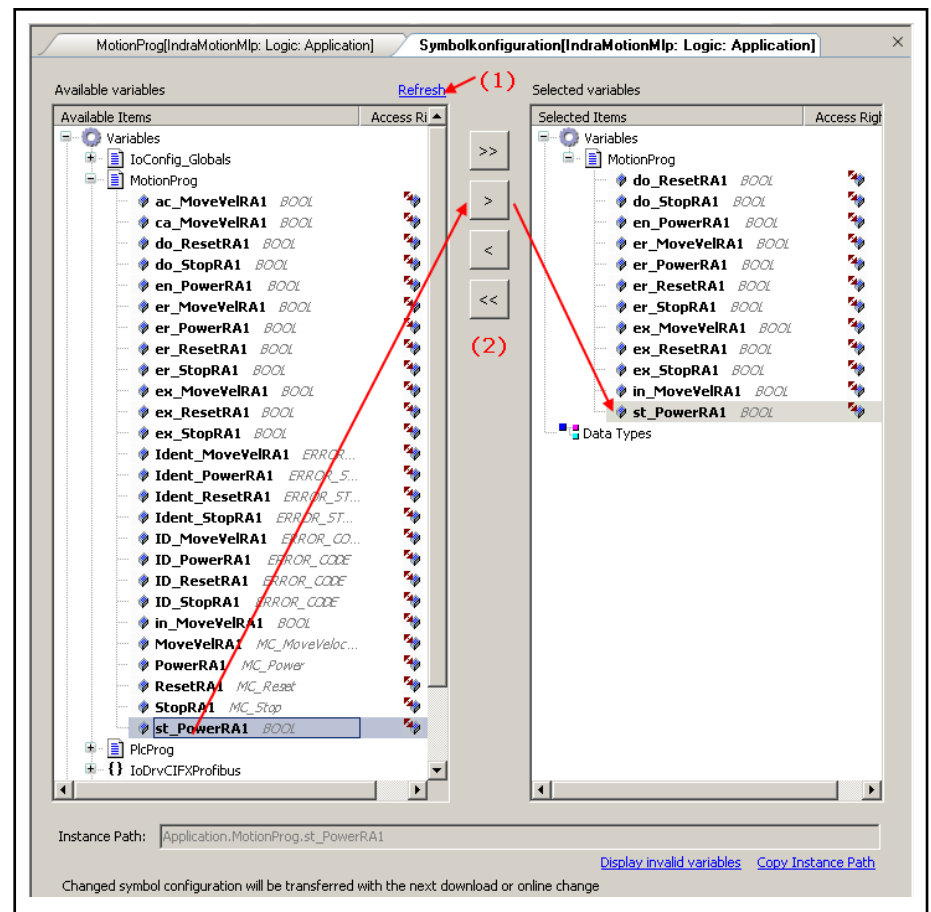


Fig.5-18: Symbol configuration – Variables selected

Three steps are required to write the symbol configuration correctly into the PLC.

1. The option **Create ► Clear All** has to be selected to delete old data in the PLC.
2. Select "Rebuild Application" subsequently since the program has changed (**Create ► Rebuild Application**).
3. Go online with the application to write the changed program into the PLC (**Debug ► Login**).

Now, the selected variables can be read by the PLC by an external application (here: WinStudio).

Drag the "VEH/VEP" visualization from the library and drop it in the project node.

Application Samples

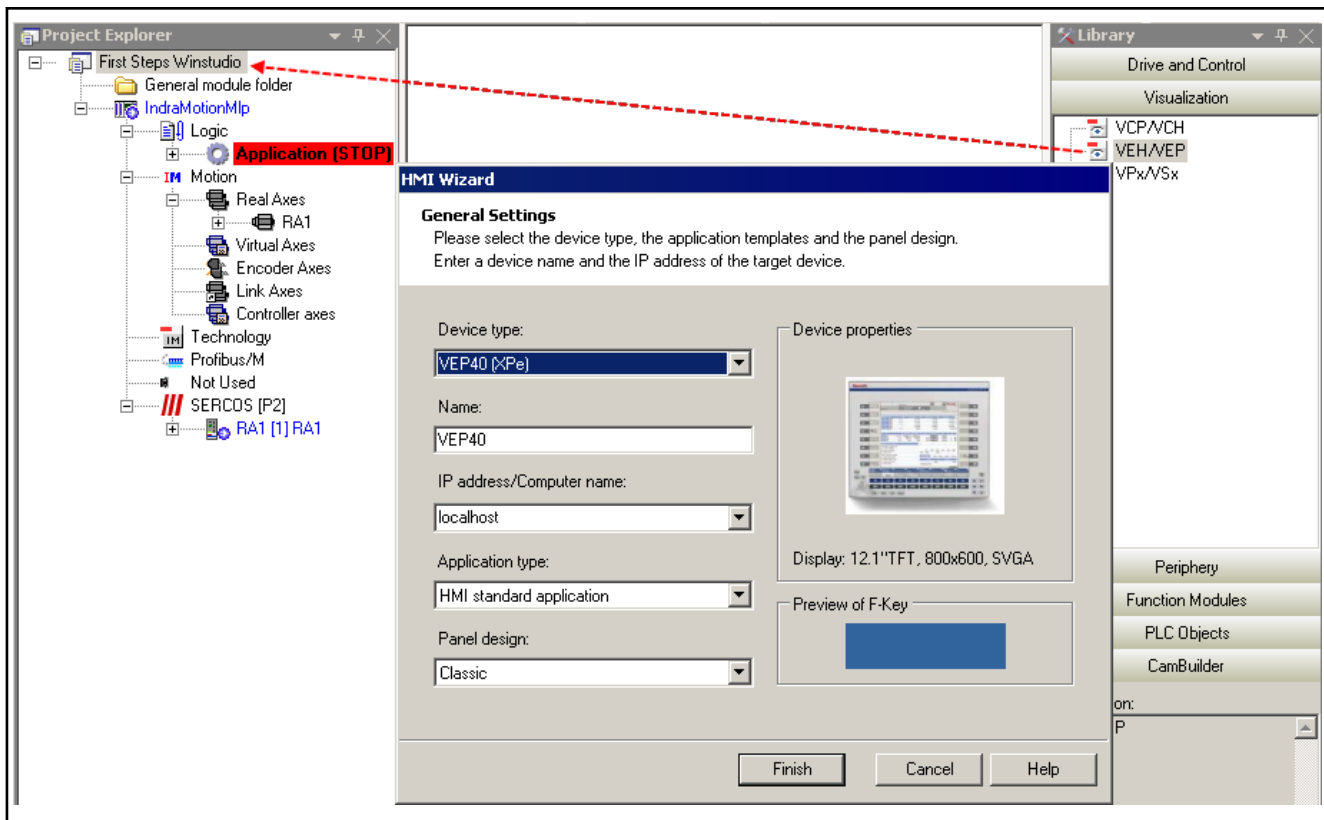


Fig. 5-19: Inserting the visualization VEH/VEP – General Settings

Select the device types in this window, specify a name for the visualization, enter the IP address of the control and select the type of application.

- The **device type** is to be selected according to the HMI (in this case VEP30CE).
- Enter the name in the **Name** field under which the visualization is executed in the project (in this case VEP30).
You can specify a name (IndraWorks will notify you if any characters chosen are invalid) or select the default name.
- Enter the IP address of the control (same you already entered when inserting the control) in the **IP address**.
- Select a WinStudio application as **Application type**.

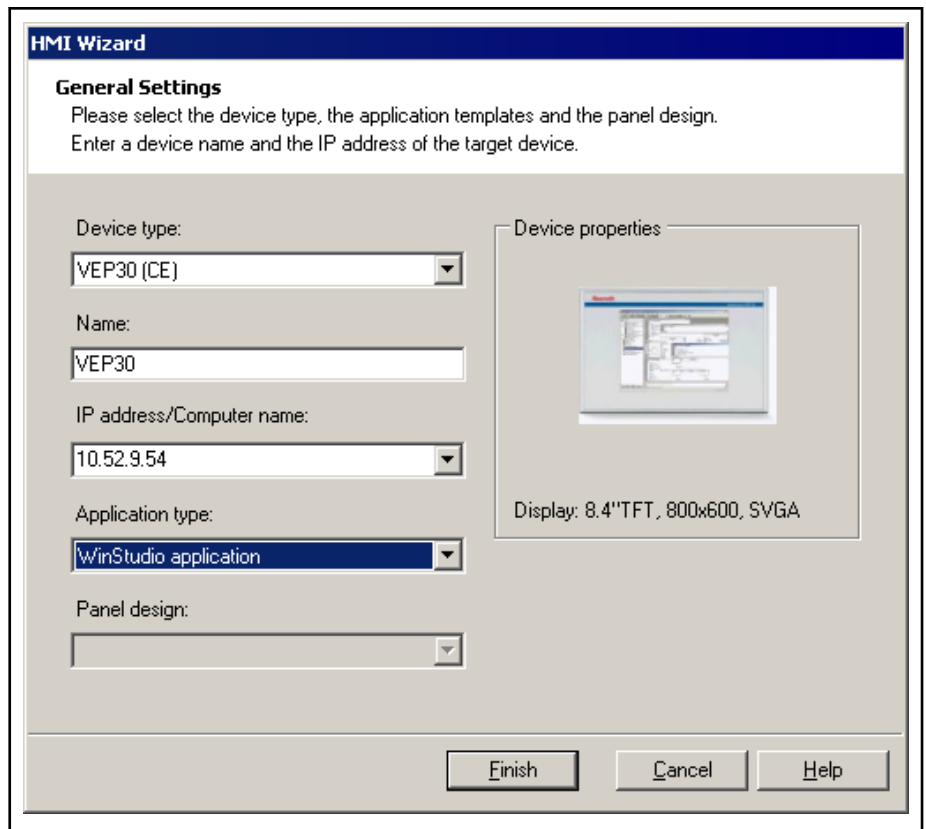


Fig.5-20: Inserting the visualization VEH/VEP – General settings selected

Now, establish the communication between WinStudio and the PLC.

Open WinStudio by clicking on the "WinStudio" object in the project tree under visualization.

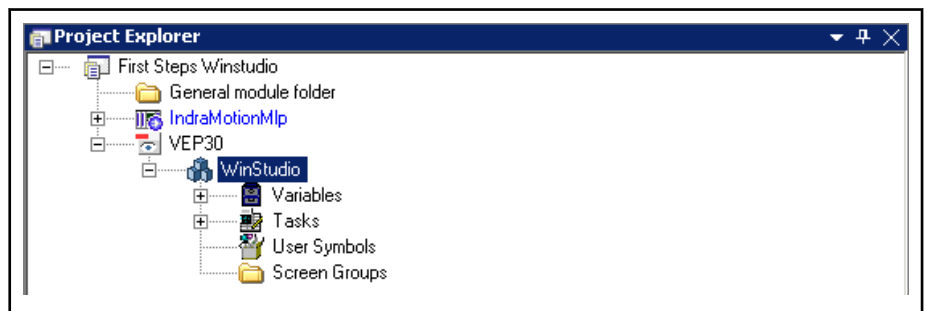


Fig.5-21: Opening WinStudio

To read the variables from the control, a special driver screen has to be integrated into WinStudio.

This driver screen is located under **VEP30 ► WinStudio ► Variables ► Communication ► BRWS ► New BR_WS driver sheet**.

Click on the BRWS driver with the right mouse button, the context menu "New BR_WS driver sheet" appears.

Application Samples

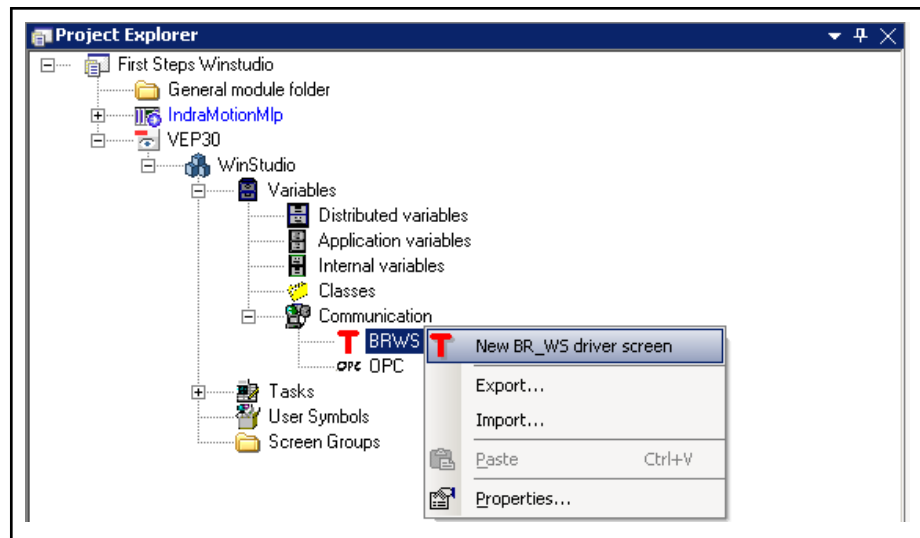


Fig.5-22: Creating a new BR_WS driver sheet

The driver must be specifically configured.

Select the "BRWS" folder with the right mouse button. Select **Properties** in the context menu.

Enter the new IP address of the control again into the window

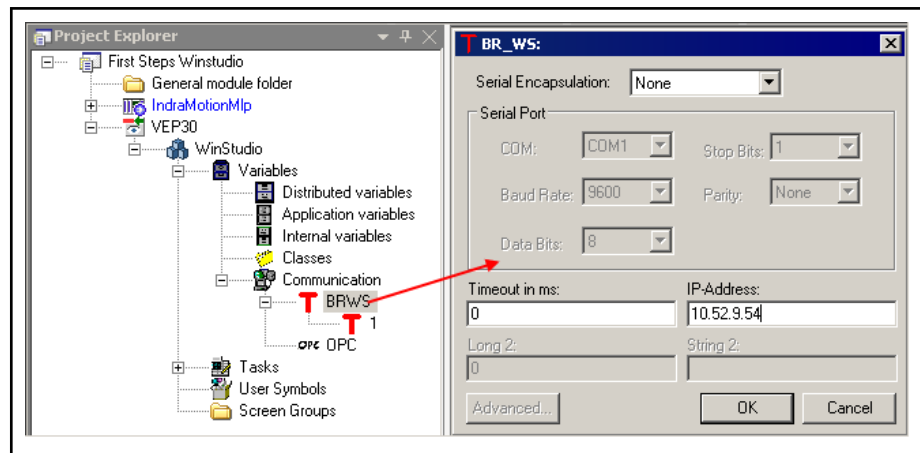


Fig.5-23: BRWS properties, here IP address

In the following step, all PLC variables of the WinStudio environment are shown.

Therefore, click on the driver sheet "1". A new window opens on the right, next to the project tree.

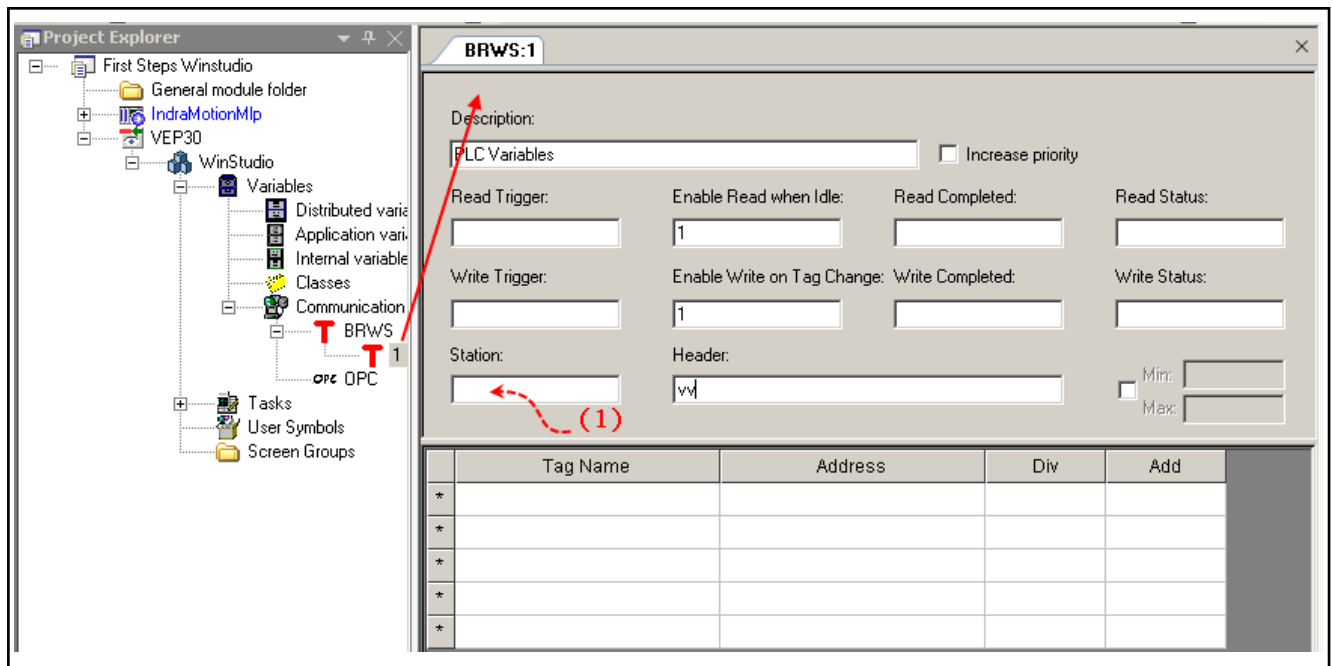


Fig.5-24: BRWS properties, here screen 1

Fill out the fields as shown above.

The fields **Enable read when idle** and **Enable write on tag change** have to assume the value 1 in order to work with the variables.

Enter something in the **Header** field such as vv.

Read the PLC variables enabled in the symbol configuration in WinStudio.

Move the mouse cursor into the **Header** and click into the **Station** field. The variable browser opens.

Select "Read Variables (online)".

Application Samples

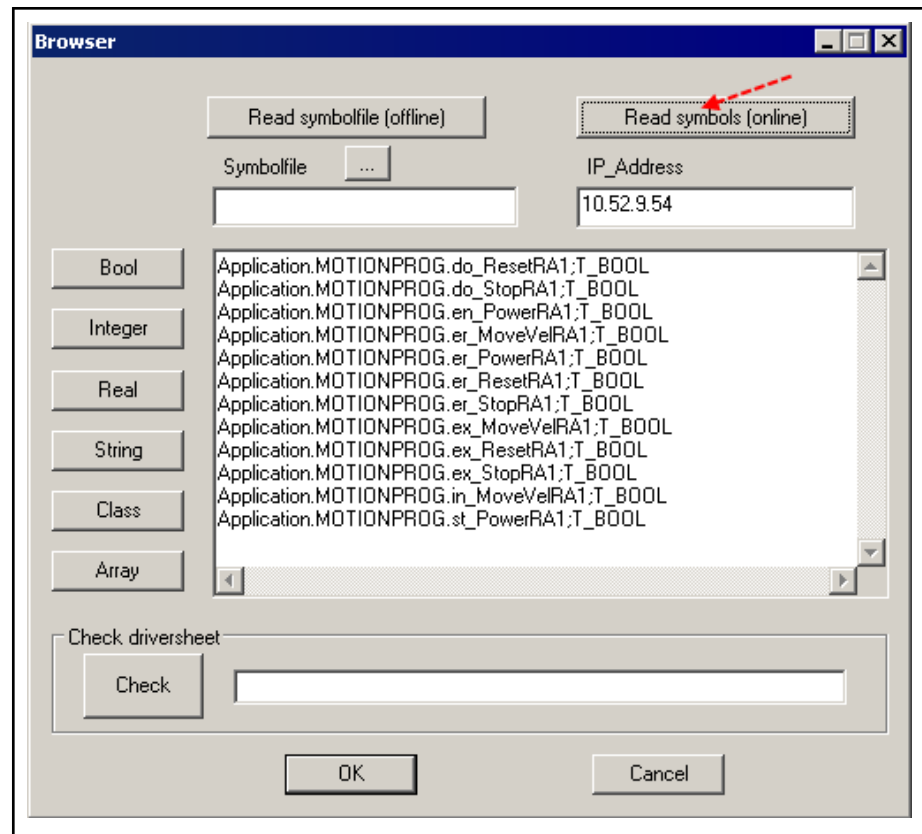


Fig. 5-25: Browser, Read variables online

All PLC variables are shown in the browser that have earlier been released under the **symbol configuration** and that can be activated via WinStudio. **PLC choice** is displayed.

The address string is specified as follows: ";" and the variable type "T_BOOL".

Highlight and copy the variables. Close the window by clicking on "OK"

Copy this variable to the "Address" (1) column, top cell.

WinStudio creates its own namespace. This is required as the visualization should be created independently and possibly before the PLC program.

Both name lists are assigned variable per variable (2).

The **Variables of WinStudio** are contained in the **Application variables** (3).

The WinStudio list can be filled in directly (Application variable list) or as shown in the following.

Application Samples

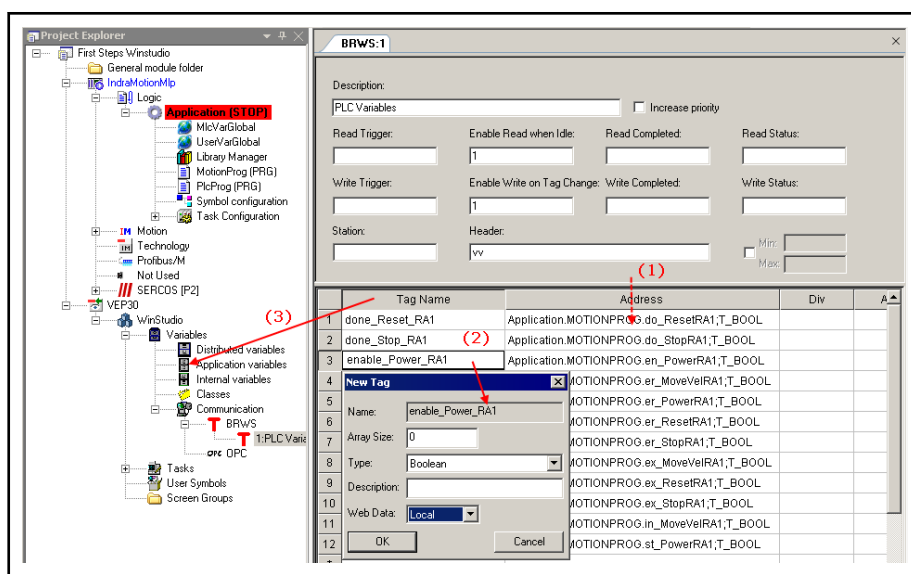


Fig.5-26: BRWS properties, here screen 1, with variable names and addresses

BRWS:1 Application variables					
	Name	Size	Type	Description	Scope
1	done_Reset_RA1	0	Boolean		Local
2	done_Stop_RA1	0	Boolean		Local
3	enable_Power_R...	0	Boolean		Local
*			Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server

Fig.5-27: WinStudio: Application variables, preliminary result

	Tag Name	Address	Div	Add
1	done_Reset_RA1	Application.MOTIONPROG.do_ResetRA1;T_BOOL		
2	done_Stop_RA1	Application.MOTIONPROG.do_StopRA1;T_BOOL		
3	enable_Power_RA1	Application.MOTIONPROG.en_PowerRA1;T_BOOL		
4	error_MoveVelocity_RA1	Application.MOTIONPROG.er_MoveVelRA1;T_BOOL		
5	error_Power_RA1	Application.MOTIONPROG.er_PowerRA1;T_BOOL		
6	error_Reset_RA1	Application.MOTIONPROG.er_ResetRA1;T_BOOL		
7	error_Stop_RA1	Application.MOTIONPROG.er_StopRA1;T_BOOL		
8	execute_MoveVelocity_RA1	Application.MOTIONPROG.ex_MoveVelRA1;T_BOOL		
9	execute_Reset_RA1	Application.MOTIONPROG.ex_ResetRA1;T_BOOL		
10	execute_Stop_RA1	Application.MOTIONPROG.ex_StopRA1;T_BOOL		
11	in_Velocity_RA1	Application.MOTIONPROG.in_MoveVelRA1;T_BOOL		
12	status_Power_RA1	Application.MOTIONPROG.st_PowerRA1;T_BOOL		
*				
*				

Fig.5-28: BRWS properties, here screen 1, with variable names and addresses

If the PLC variables have been integrated correctly, all variables are listed under

WinStudio ► Variables ► Application Variables .

Application Samples

BRWS:1 Application variables					
	Name	Size	Type	Description	Scope
1	done_Reset_RA1	0	Boolean		Local
2	done_Stop_RA1	0	Boolean		Local
3	enable_Power_R...	0	Boolean		Local
4	error_MoveVelocit...	0	Boolean		Local
5	error_Power_RA1	0	Boolean		Local
6	error_Reset_RA1	0	Boolean		Local
7	error_Stop_RA1	0	Boolean		Local
8	execute_MoveVel...	0	Boolean		Local
9	execute_Reset_R...	0	Boolean		Local
10	execute_Stop_RA1	0	Boolean		Local
11	in_Velocity_RA1	0	Boolean		Local
12	status_Power_RA1	0	Boolean		Local
*			Integer		Server

Fig.5-29: Checking the transfer: WinStudio – Application variables

5.4.5 Creating the WinStudio Visualization

After the variables have been integrated and are now available as user variables, you can start with the graphic interface.

Close the current window. Select the "WinStudio" folder with the right mouse button. Insert a new screen in the next step.

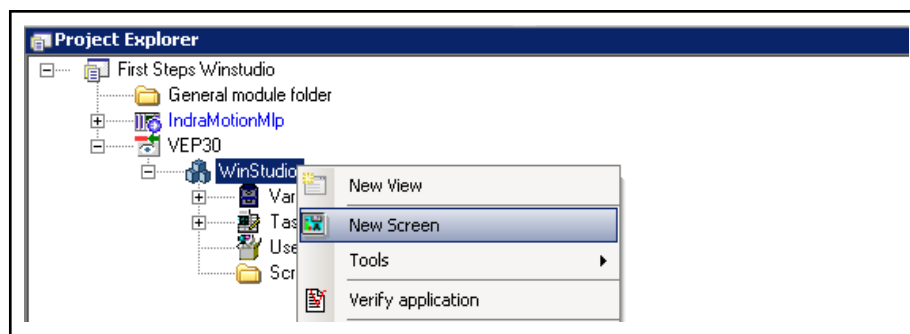


Fig.5-30: WinStudio – Insert screen

You can freely select the name of the screen: here "start screen".

Open the screen. The "Screen Attributes" can be selected after clicking on the screen with the right mouse button.

Application Samples

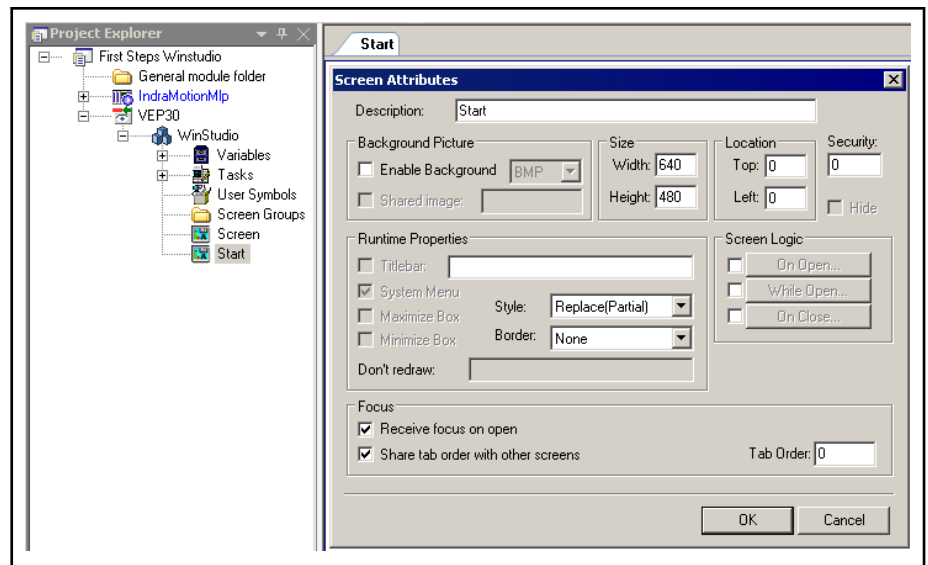


Fig.5-31: WinStudio – Screen/Settings

The workspace next to the tree opens as well. Objects can be inserted into the screen using the individual toolbars at the right WinStudio margin.

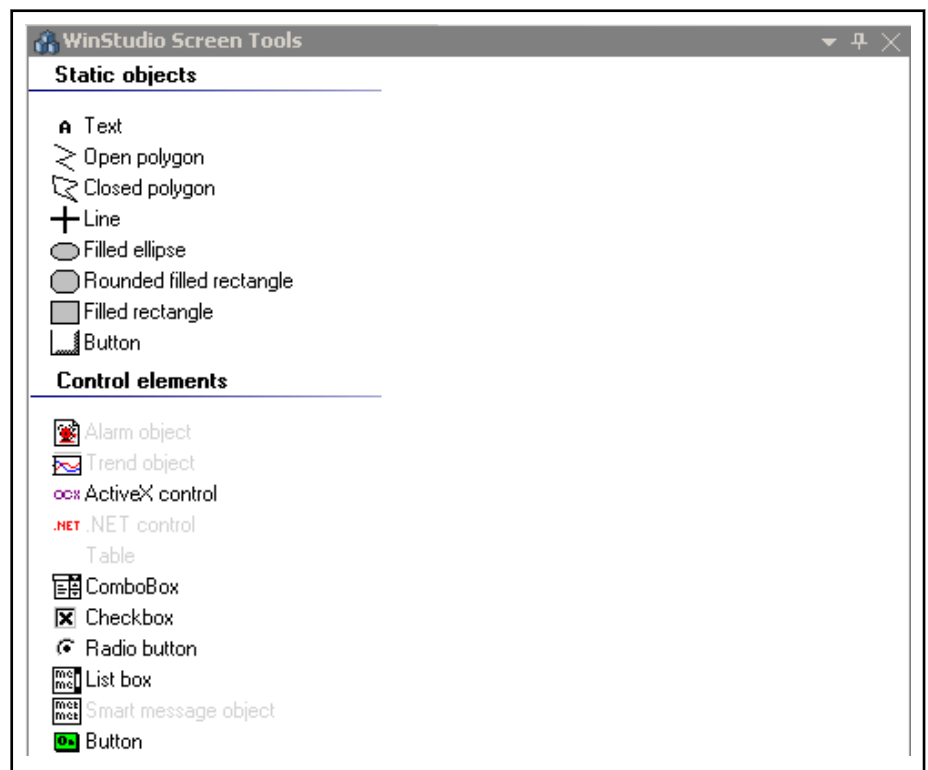


Fig.5-32: WinStudio – Toolbox

The objects are prepared for a small demonstration interface. Therefore, change the background color of the later interface with the right mouse button.

Application Samples

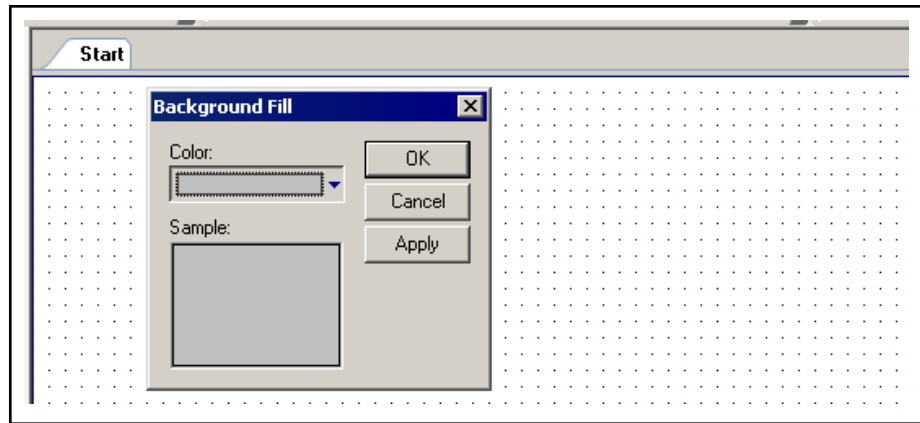


Fig.5-33: "Startup" screen - Change background color

To divide the interface optically and to learn about the elements of the tool-box, add the "Filled Rectangle" object to the screen. This rectangle should cover the upper quarter of the work space.

The object can be adapted more specifically if you open the context menu with the right mouse button. Select "Properties".

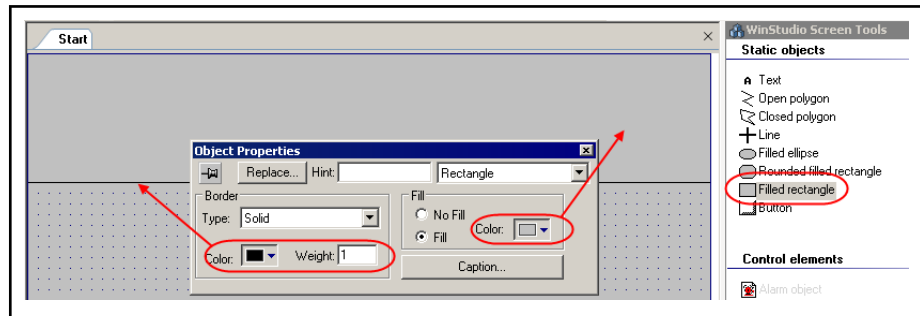


Fig.5-34: WinStudio – Insert the "Filled rectangle" object

As shown, the filling color was adapted and the rectangle was cut to the complete screen length.

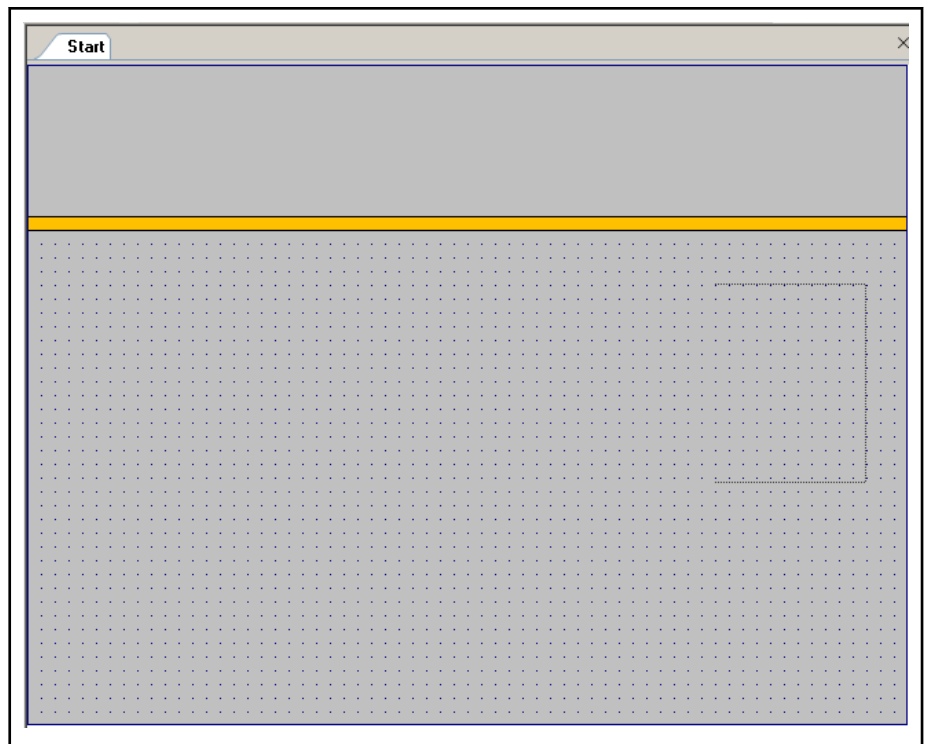


Fig.5-35: WinStudio – Add another object of type "Filled rectangle"

WinStudio provides the user with a library containing symbols for example.

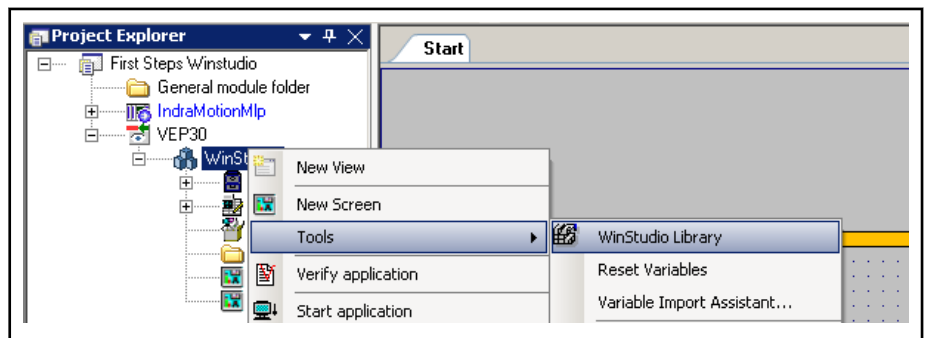


Fig.5-36: WinStudio – Library selected

The library can be accessed via the main menu "WinStudio" or via the context menu of the WinStudio folder.

Application Samples

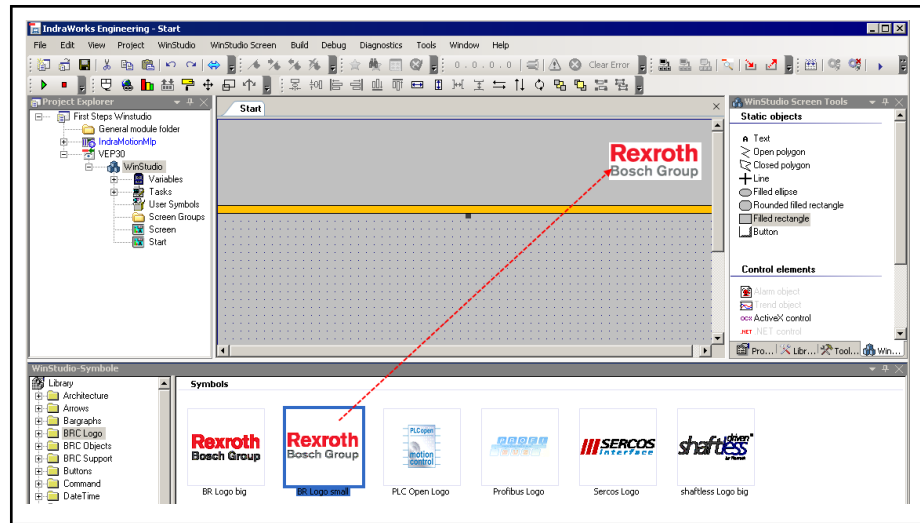


Fig.5-37: WinStudio – Library opened, logo applied

The requested symbol, here "BR Logo small", is inserted via drag&drop.

Now, insert a text object from the toolbar into the screen. Formatting can be selected via the object settings (right mouse click ► **Settings**).

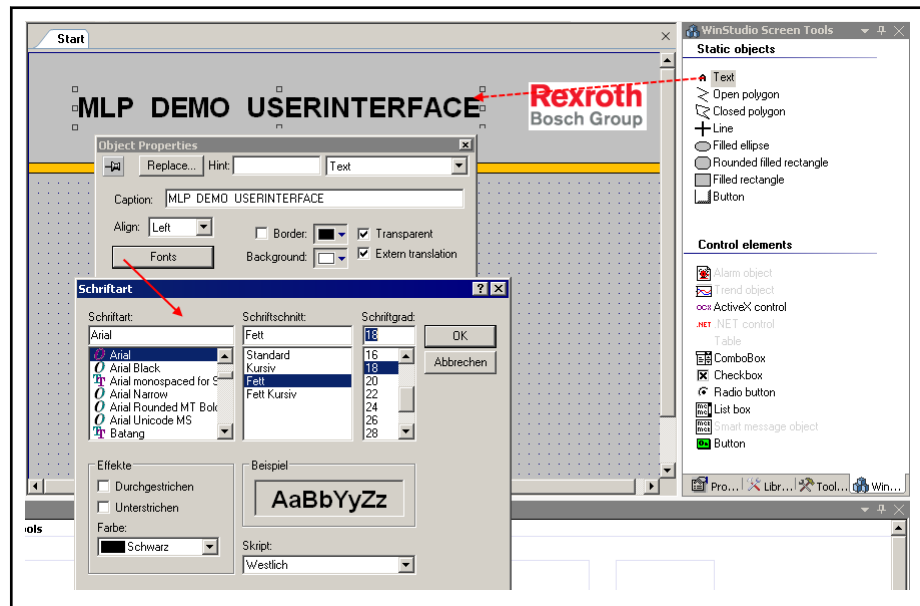


Fig.5-38: WinStudio – Insert text object

The control elements are now inserted into the screen. A pushbutton (control element "button") contained in the toolbar is used as an example.

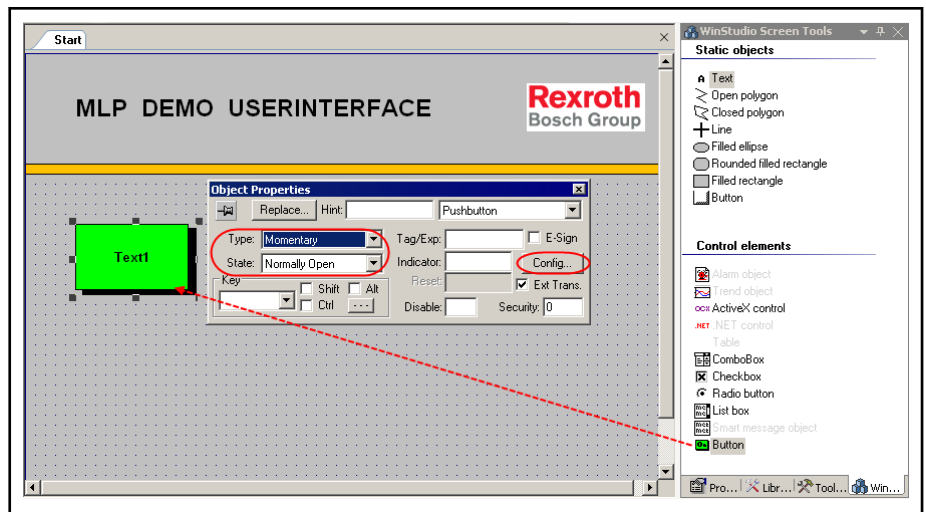


Fig.5-39: WinStudio – Pushbutton inserted

Different settings have to be made for the control elements.

As shown in the figure above, the highlighted fields must be set.

Configuring the pushbutton as **type "Switch"**.

Insert the variable to be activated via the button in the **Tag/Exp** field by double-clicking in this field

(in this case **enable_Power_RA1**).

The **"Object Finder"** to select the variable opens.

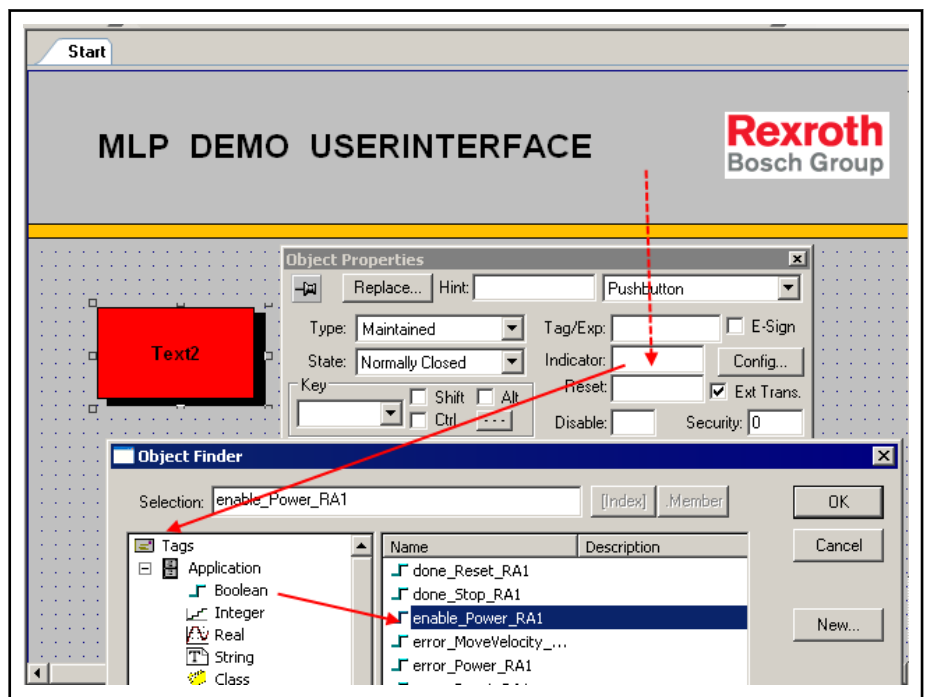


Fig.5-40: Variable settings / tag via object finder

Set the behavior in case of an open and closed switch under **Configuration**.



The settings "Color selection", "Status" (normally closed/ normally open) and the "Text" are to be assigned user-specifically.

Application Samples

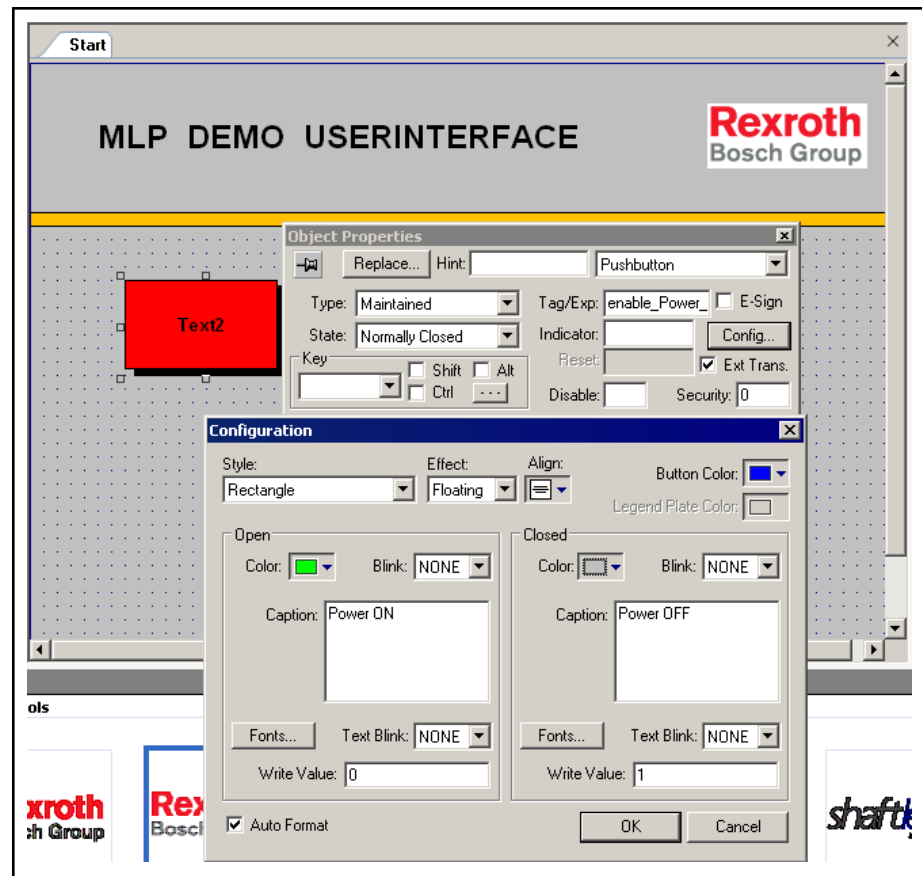


Fig.5-41: Settings regarding type, state and configuration

The 4 pushbuttons with the following configurations have been implemented:

Connect power

- Type: Switch
- Status: Normally closed
- Variable: enable_Power_RA1
- Label (open): Power ON / green
- Label (closed): Power OFF / grey

Reset drive

- Type: Switch
- Status: Normally closed
- Variable: execute_Reset_RA1
- Label (open): Reset drive / green
- Label (closed): Reset OFF / grey

Switch on drive

- Type: Switch
- Status: Normally closed
- Variable: execute_MoveVelocity_RA1
- Label (open): Drive RUN / green
- Label (closed): Drive ON / grey

Stop drive

- Type: Switch

- Status: Normally closed
- Variable: execute_Stop_RA1
- Label (open): Drive OFF / green
- Label (closed): Drive STOP / grey

To display the current status of the drive, add checkboxes to the interface.
Make the settings **"Description"** and **"Variable"**.

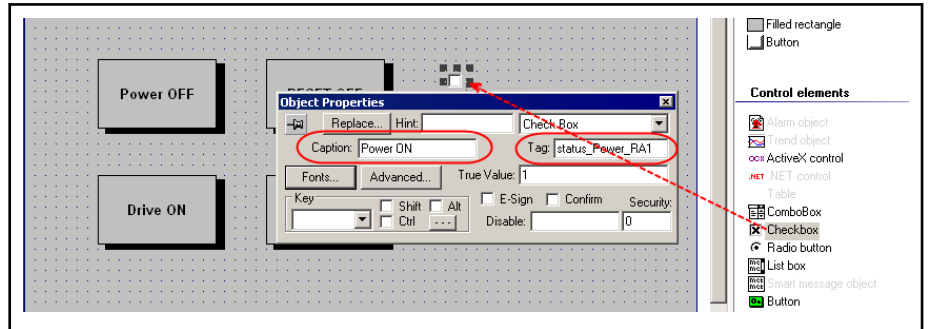
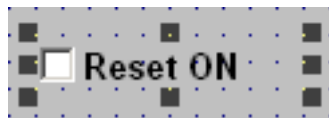


Fig. 5-42: WinStudio – Insert checkbox



To show the label, please adjust the size of the element.



See the configuration of all checkboxes below:

Power status

- Description: Power ON
- Variable: status_Power_RA1

Power status (error)

- Description: Power ERROR
- Variable: error_Power_RA1

Drive status

- Description: Drive ON
- Variable: in_Velocity_RA1

Drive status (error)

- Description: Drive ERROR
- Variable: error_MoveVelocity_RA1

Status reset drive

- Description: Reset ON
- Variable: done_Reset_RA1

Application Samples

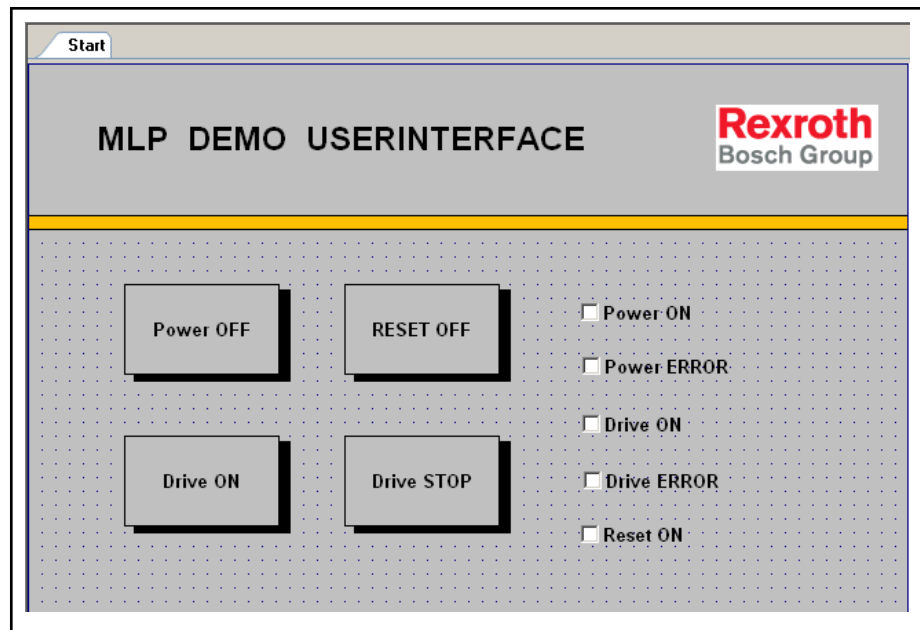


Fig.5-43: WinStudio – Complete interface

5.4.6 Starting the Interface

Just a few steps are left and the interface is running on the control.

First, set the WinStudio Viewer **"Automatic"** for startup. This displays the interface on the HMI (under Windows CE).

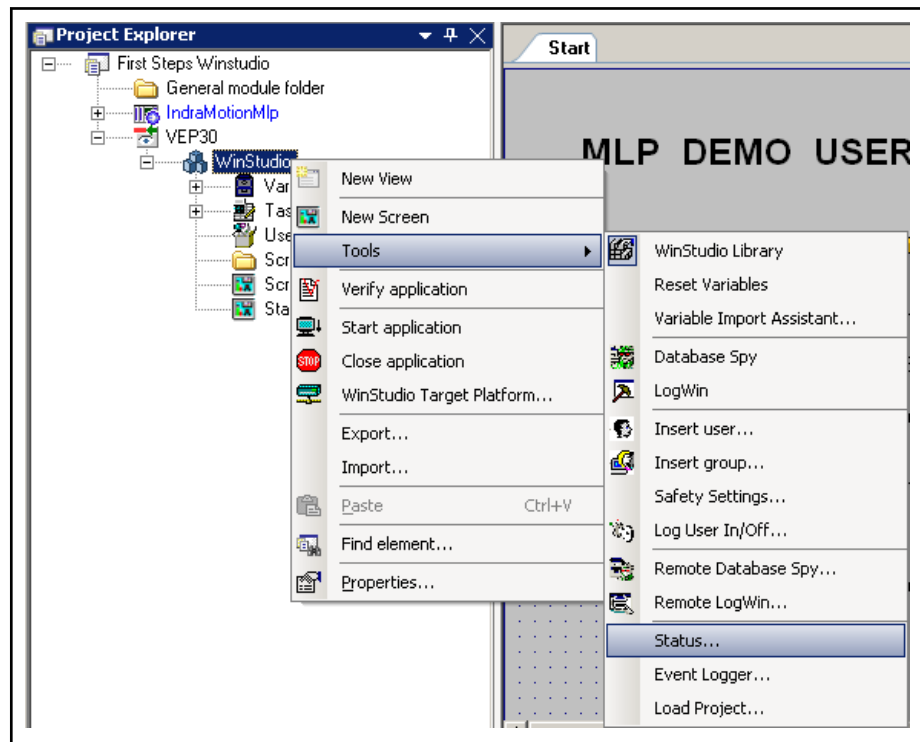


Fig.5-44: WinStudio – Project / Select status

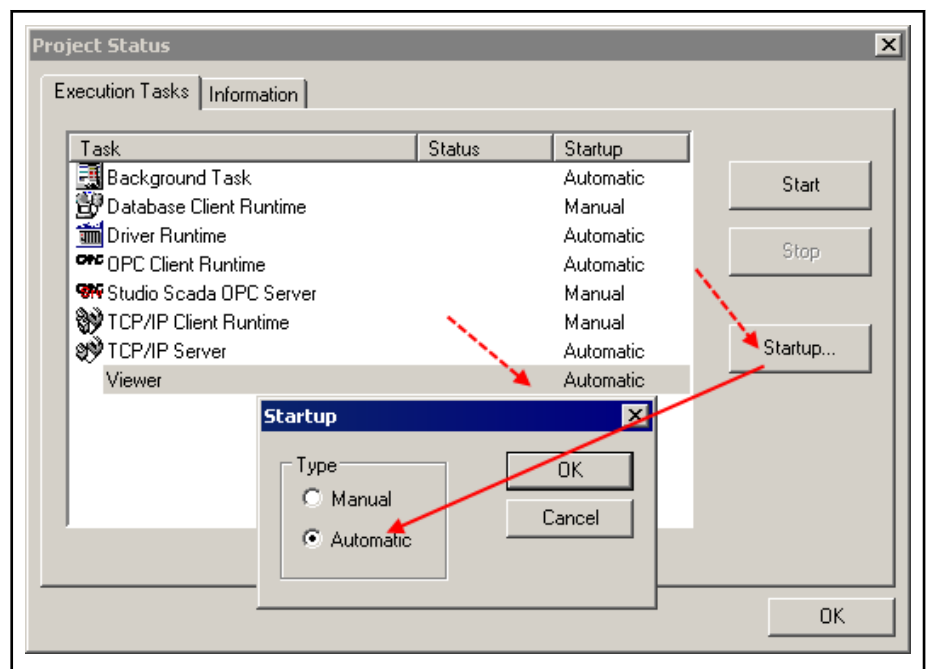


Fig. 5-45: Project settings – Viewer startup "Automatic"

In the next step, select an important option in the project settings.

Select the start screen you created in the field **"Start screen"** under **WinStudio ► Properties ► Runtime Desktop** (otherwise, the interface is not loaded in the Viewer).

You can also select in this tab, whether the initial Windows CE toolbar is shown or hidden as it can be done with the toolbar of the WinStudio Viewer.

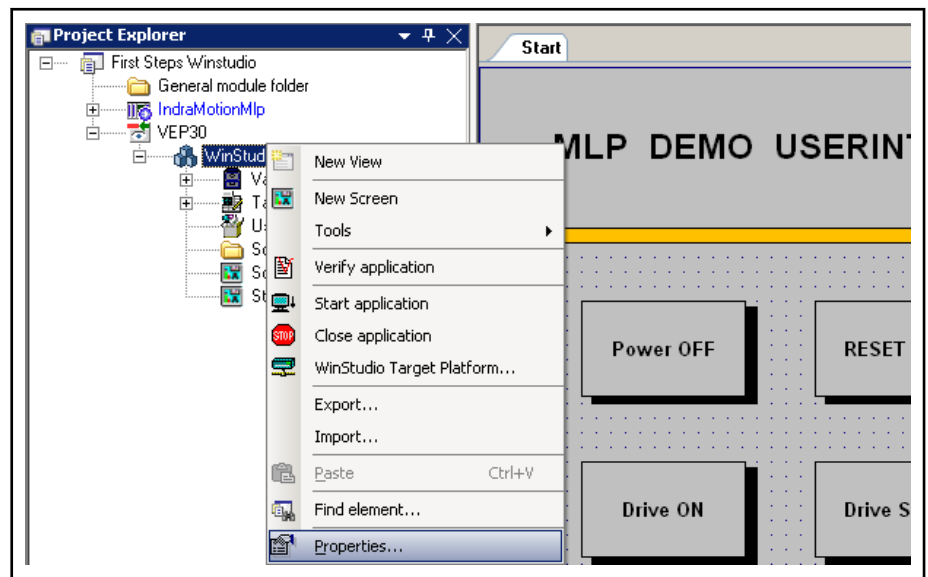


Fig. 5-46: WinStudio – Project/ Select properties

Application Samples

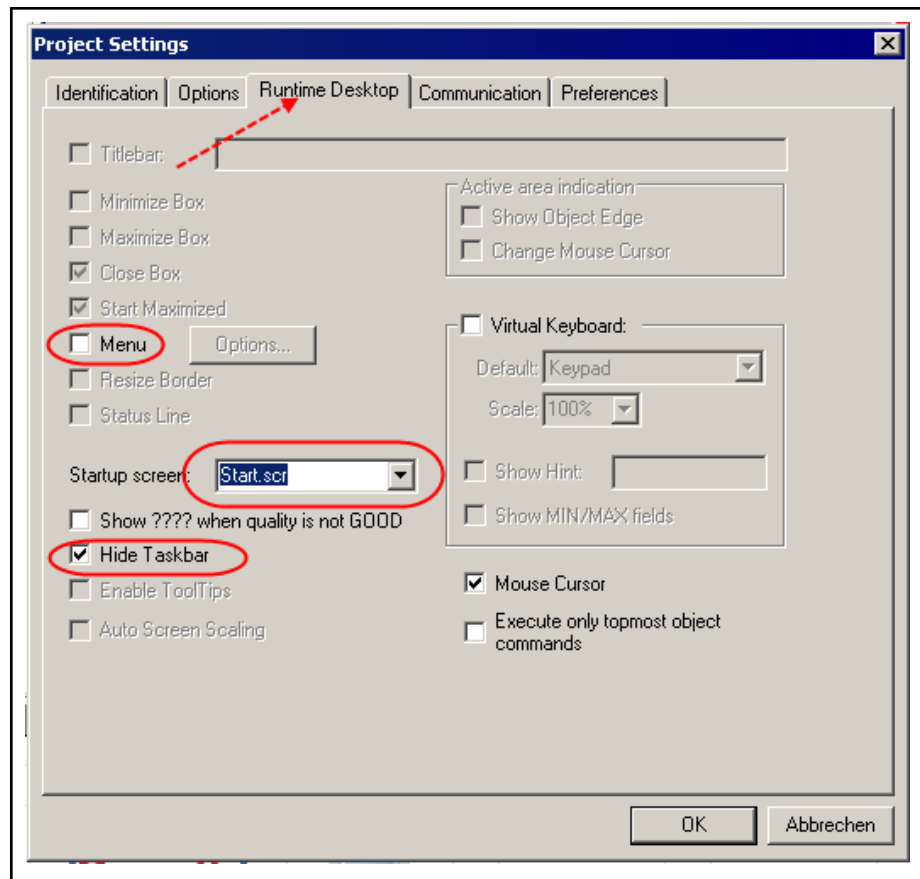


Fig.5-47: WinStudio – Project Settings

The settings made can now to be saved in the project.

Transfer your PLC project in a ready for operation state:

1. Go online with the control.
2. Log in.
3. Start your application.
4. Switch the sercos bus to phase 4 (P4).

Transfer the WinStudio project to the target platform.

Select the context menu item WinStudio target platform

Application Samples

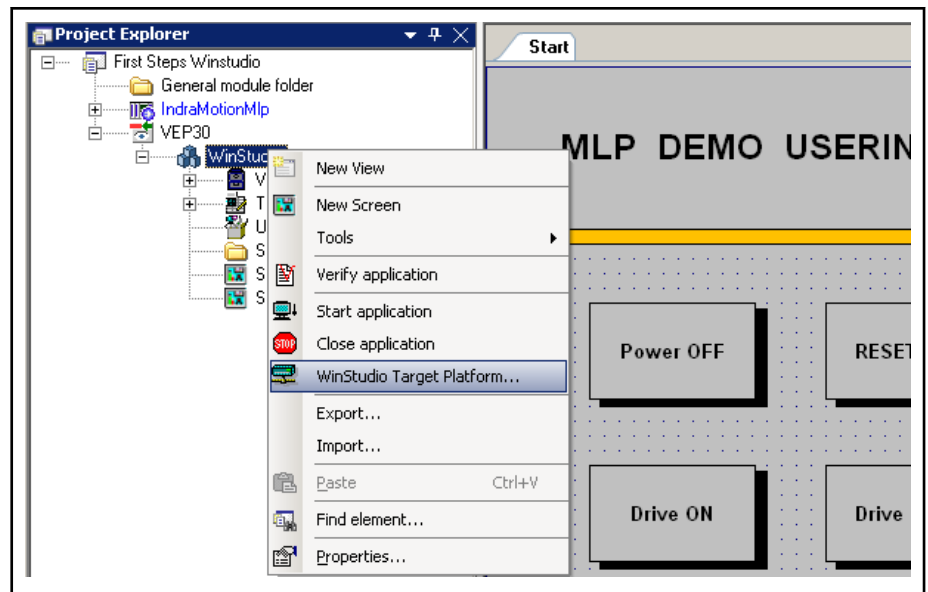


Fig.5-48: Context menu item: WinStudio target platform

Connect the interface to the HMI (1) and install the system (2).

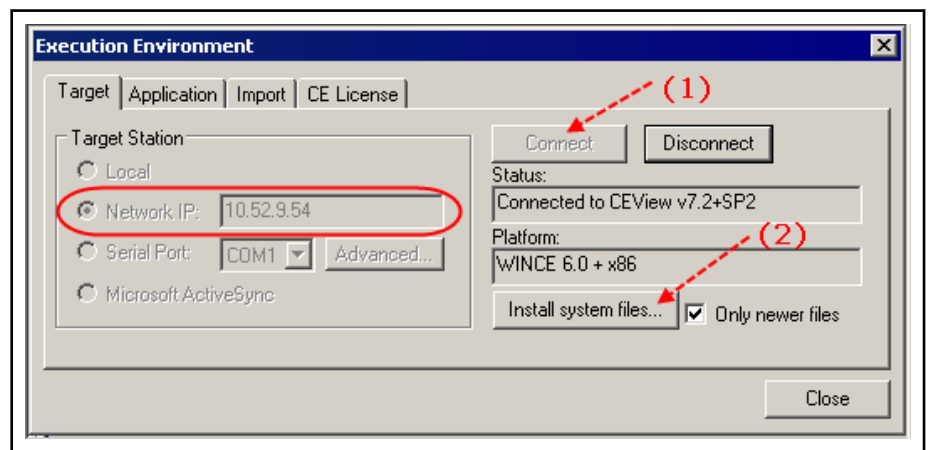


Fig.5-49: Execution platform: Installation system

Subsequently, load the project (1) (the loading process is displayed on the VPPscreen).

Start the application (2).

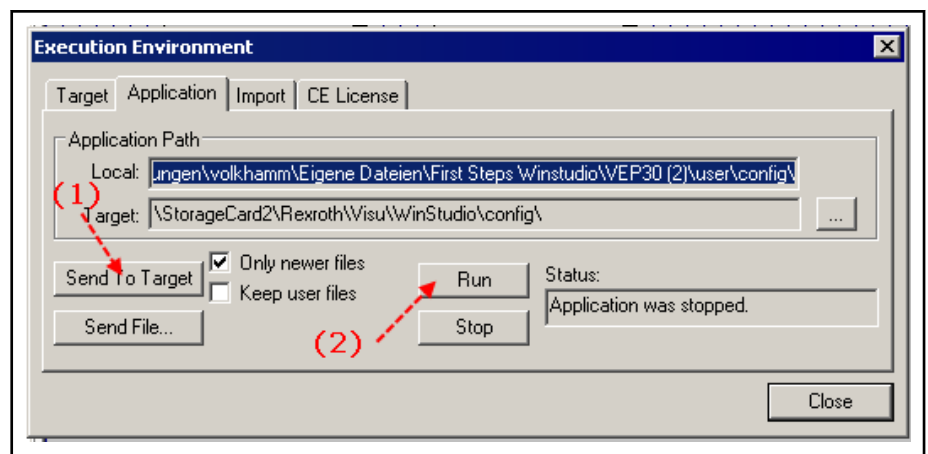


Fig.5-50: Loading and starting the application

Application Samples

Start interface automatically when booting the control

The project is now contained in the control.

To automatically start the interface (WinStudio application) when booting Windows XP, set the following option:

In Windows XP, open **Start ► Programs ► Tools ► Rexroth Settings**.

Select the **"Application Settings"** tab.

Select the menu item **"WinStudio"** in the **"Start App"** field under **"Visualization"**.

When closing the Rexroth settings, the user is requested to reboot the control. The change becomes active after a reboot and the interface starts automatically.

5.5 IndraLogic Visualization

An application sample for the VI-Logic visualization is contained in "Rexroth IndraLogic XLC 12VRS First Steps", DOK-XLC***-F*STEP**V12-CO01-EN-P.

5.6 HMI of 3rd Party Manufacturers

5.6.1 OPC.DA Server

Tools to Configure the OPC.DA Server on a PC

The program listings ["Batch processing program for opening and editing the file OPCServer.ini" on page 160](#) and ["Batch processing program for opening the file OPCServer.log" on page 160](#) can be used for a more user-friendly handling of the OPC.DA server configuration file "OPCServer.ini" as well as the "OPCServer.log" log file on the PC. The file paths can be determined using the registry and the environment variables. Open them in the "Notepad" editor ("SET EDITOR_FILE=notepad.exe").

Batch processing program for opening and editing the file "OPCServer.ini"

```
@ECHO OFF
REM
REM Open IndraLogic OPC Server Configuration File.
REM If it does not exist, copy it!
REM

SET EDITOR_FILE=notepad.exe
SET REGKEY=HKLM\Software\Rexroth\IndraWorks
SET REGKEYNAME=CurrentIndraLogicOPCDAIniFileName

REM 'delims' is set with TAB and SPACE
FOR /F "tokens=2* delims=" %%a IN ('REG QUERY "%REGKEY%" /v %REGKEYNAME%') DO SET OPCFILE="%%b"

IF EXIST "%OPCFILE:="%" GOTO fileopen

COPY "%CommonProgramFiles%\Rexroth\OPCServer.ini.template" "%OPCFILE:="%"

:fileopen
START %EDITOR_FILE% "%OPCFILE:="%
```

Fig.5-51: *Batch processing program for opening and editing the file "OPCServer.ini"*

Batch processing program for opening the file "OPCServer.log"

```
@ECHO OFF
REM
REM Open IndraLogic OPC Server Log File.
REM

SET EDITOR_FILE=notepad.exe
SET REGKEY=HKLM\Software\Rexroth\IndraWorks
SET REGKEYNAME=CurrentIndraLogicOPCDALogFileName
```

```

REM 'delims' is set with TAB and SPACE
FOR /F "tokens=2* delims=" "%%a IN ('REG QUERY "%REGKEY%" /v %REGKEYNAME%') DO SET OPCFILE="%%b"

IF EXIST "%OPCFILE:=" GOTO fileopen

GOTO :EOF

:fileopen
START %EDITOR_FILE% "%OPCFILE:="

```

Fig. 5-52: Batch processing program for opening the file "OPCServer.log"

Tools to Transfer the OPC.DA Server Configuration to a Visualization Device VEPxx.4 with WindowsCE

If the visualization device ("VEPxx.4") is in the same network as the development PC the script ["Batch processing program for copying the file OPCServer.ini to a VEPxx.4" on page 161](#) can be used to transfer the OPC.DA server configuration file "OPCServer.ini" which is active on the PC.

Adjust the IP address in the following assignment line:

"SET IPADDR_VEP=192.168.1.11"

Batch processing program for copying the file "OPCServer.ini" to a "VEPxx.4"

```

@ECHO OFF
CLS
REM
REM Copy IndraLogic OPC Server Configuration File to VEPxx.4
REM
REM
REM Set IP Address of the VEP Device
SET IPADDR_VEP=192.168.1.11

SET REGKEY=HKLM\Software\Rexroth\IndraWorks
SET REGKEYNAME=CurrentIndraLogicOPCDAIniFileName

REM 'delims' is set with TAB and SPACE
FOR /F "tokens=2* delims=" "%%a IN ('REG QUERY "%REGKEY%" /v %REGKEYNAME%') DO SET OPCFILE="%%b"

IF EXIST "%OPCFILE:=" GOTO filecopy

ECHO.
ECHO !!! ERROR !!!
ECHO - File
ECHO %OPCFILE%
ECHO does not exist!
ECHO.

PAUSE
GOTO :EOF

:filecopy

REM Create FTP command file
SET FTPCMDFILE="%~dp0FtpCmd.txt"
SET VEPDSTDIR=/StorageCard/Rexroth/Controls/OPCServer

ECHO open %IPADDR_VEP%> %FTPCMDFILE%
ECHO anonymous>> %FTPCMDFILE%
ECHO 123@456.com>> %FTPCMDFILE%
ECHO cd %VEPDSTDIR%>> %FTPCMDFILE%
ECHO ascii>> %FTPCMDFILE%
ECHO put "%OPCFILE:=">> %FTPCMDFILE%
ECHO close>> %FTPCMDFILE%
ECHO bye>> %FTPCMDFILE%

ECHO.
ECHO Copy file
ECHO %OPCFILE%
ECHO.
ECHO.
CALL ftp -s:%FTPCMDFILE%

```

Application Samples

PAUSE

Fig. 5-53: Batch processing program for copying the file "OPCServer.ini" to a "VEPxx.4"

5.6.2 ModbusTCP

General

It is useful for the configuration to execute first tests with the Modbus test clients.

Test client examples available in the internet for free (date 09.2008):

- IPC@CHIP Modbus Client V1.4
- Modbus TCP/IP Request Tool Pro

The following chapters describe the configuration of a TCP connection with

Details about the configuration of a Modbus TCP connection is explained in the following chapters:

- [Configuration with Siemens WinCC-flexible 2007, page 163](#) and
- [Configuration with Pro-face GP-Pro EX 2.2, page 167](#)

Addressing data from a PLC project

In the PLC project, data ranges for the register and bit data are transferred to function block instance IL_ModbusTCPServer using the pointer.



The data ranges of the Modbus TCP server are accessed via the following addressing by the Modbus client

Register data range:

Addressing the "Holding Register" data using **%MW** (read and write access) or the "Input Register" data using **%IW** (only read access).

Bit data range:

Addressing the "Coil" data using **%M** (read and write access) or the "Discrete Input" data using **%I** (only read access).

Bit data

In the test project, the bit data range was assigned as array of bytes from where every single bit can be addressed continuously.



No ARRAY OF BOOL can be used for the bit data range since 8 bits of memory are assigned to a variable of BOOL in the IndraLogic.

The individual bits of a byte can be copied on variables of type BOOL (and vice versa)

The address of a variable from a bit data range corresponds to the bit number from the byte array used for bit data.

Example:

Bit data (%M)

%M0 address corresponds to the **BitData[0].0** variable

%M7 address corresponds to the **BitData[0].7** variable

%M8 address corresponds to the **BitData[1].0** variable

%M12 address corresponds to the **BitData[1].4** variable

Register data The register data range can consist of up to 65536 words and can be organized by the user without any restrictions.

For a better overview on the register data, a structure containing arrays of different PLC data types was used for the register data range in the test project. This structure was created in such a way that the individual data type areas are always provided with a length of 1000 words and do therefore begin with a defined word address.

Example of a data structure for RegisterData

Program:

```

TYPE REGISTER_DATA :
STRUCT
  arWord:   ARRAY[0..999]OF WORD;      (* %MW0000 - %MW0999 *)
  arInt:    ARRAY[0..999]OF INT;       (* %MW1000 - %MW1999 *)
  arUInt:   ARRAY[0..999]OF UINT;      (* %MW2000 - %MW2999 *)
  arDint:   ARRAY[0..499]OF DINT;      (* %MW3000 - %MW3999 *)
  arUdint:  ARRAY[0..499]OF UDINT;     (* %MW4000 - %MW4999 *)
  arReal:   ARRAY[0..499]OF REAL;     (* %MW5000 - %MW5999 *)
  arString: ARRAY[0..99]OF STRING(19); (* %MW6000 - %MW6999 *)
(* String(19): because 19 byte string + 1 byte end sign *)
(* together 7000 WORDS *)
END_STRUCT
END_TYPE

```

Text string Texts (strings) can also be transferred via the register data area.



In the IndraLogic, the length of a string (in byte) is always the given length + 1 byte of end ID

To define the strings with their total word length, the length is always to be specified with an odd number of bits.

(e.g. string(19) => 19 byte string + 1 byte End ID = 20 byte = 10 words)

The address of a variable from the register data range results from the design of the structure used for the register data.

If the previously shown ["Example of a data structure for RegisterData" on page 163](#) is used, the following addresses result:

Example:

Register data (%MW)

%MW0 address corresponds to the **RegisterData.arWord[0]** variable

%MW1000 address corresponds to the **RegisterData.arInt[0]** variable

%MW1001 address corresponds to the **RegisterData.arInt[1]** variable

%MW5000 address corresponds to the **RegisterData.arReal[0]** variable

%MW5002 address corresponds to the **RegisterData.arReal[1]** variable

%MW6000 address corresponds to the **RegisterData.arString[0]** variable

%MW6010 address corresponds to the **RegisterData.arString[1]** variable

Configuration with Siemens WinCC-flexible 2007

Communication settings

To configure the connection in WinCC-flexible, the following settings are required:

1. The following menu is entered in the project tree of the operating device
Communication ► Connections
2. Enter the name of the connection (and optionally a comment) and select the communication driver **"Modicon MODBUS TCP/IP"**
3. Subsequently, the communication parameters have to be configured in the dialog

Application Samples

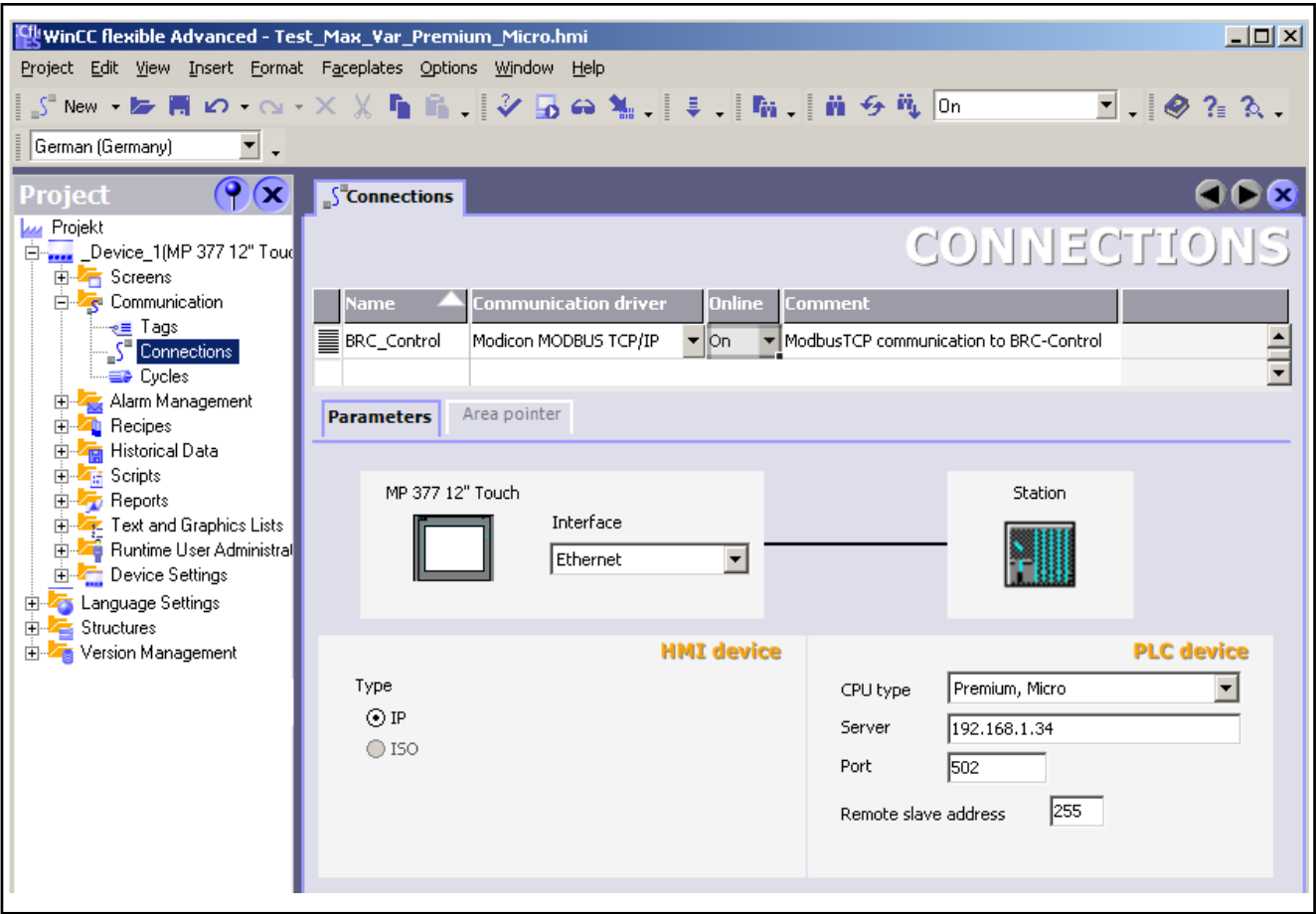


Fig.5-54: WinCC-flexible Modbus TCP communication parameters

Section	Name	Value	Description
HMI device	Interface	Ethernet	Selection of the communication interface
	Type	IP	Only IP is supported
PLC device	CPU type	Premium, Micro	Schneider Modicon control type operating without address offset and regular bit counting (refer to the Siemens user manual "WinCC-flexible 2007 Communication Volume 2 - Chapter 6"
	Server	"192.168.1.34"	IP address of the control
	Port	"502"	Port of the communication connection (port of the IL_ModbusTCPServer function block instance)
	Remote slave address	"255"	Maintain default setting (255) ! not relevant for the Modbus TCP - refer to "MODBUS Messaging on TCP/IP Implementation Guide V1.0b" on www.Modbus-IDA.org

Fig.5-55: WinCC-flexible Modbus TCP communication parameters

Variable Declaration After configuring the connection, the variables can be declared. Under the project tree of the operating device, open the menu **Communication ▶ Tags**.
Example of a variable declaration in WinCC-flexible 2007:

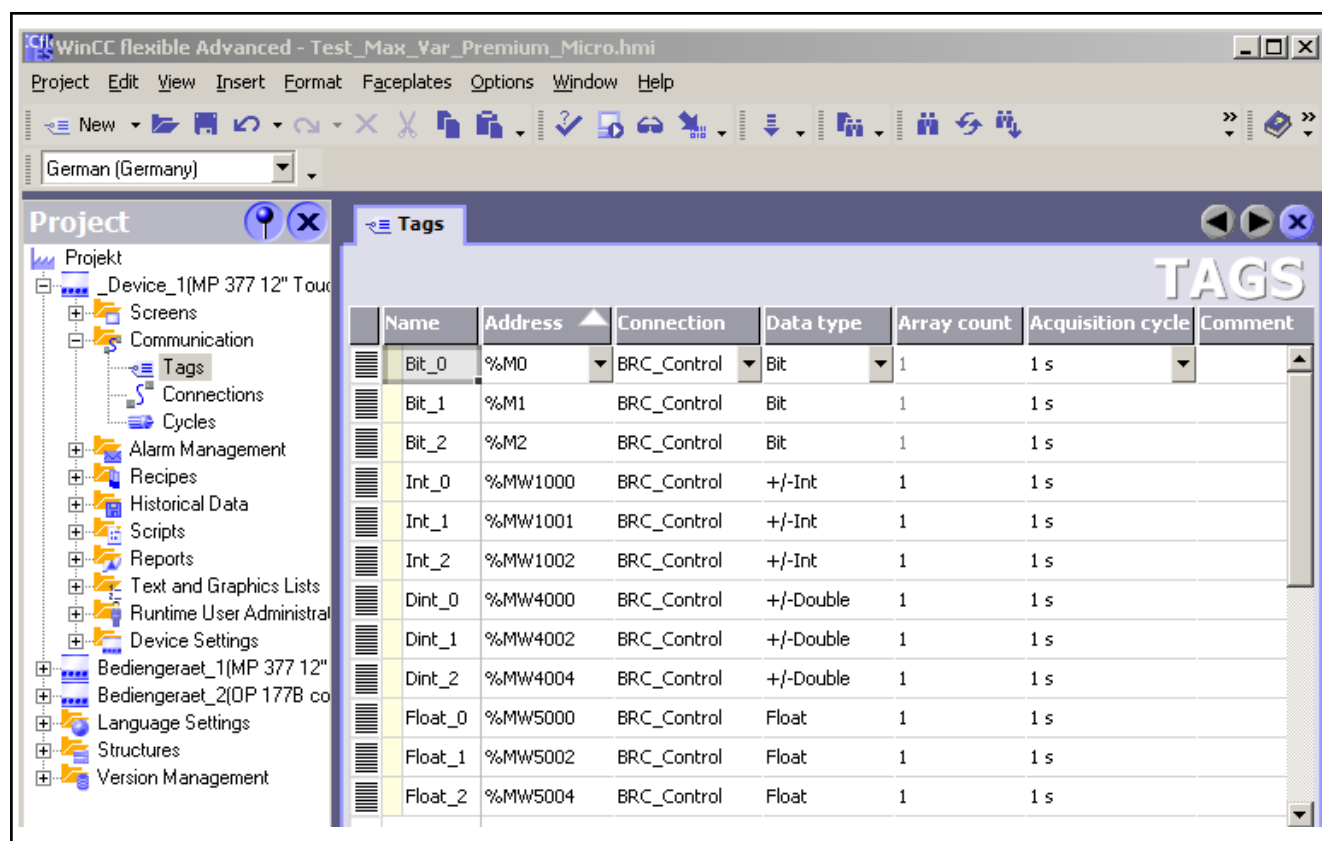


Fig. 5-56: WinCC-flexible variable declaration - Example

WinCC - flexible variable declaration

Input field	Example	Description
Name	"Bit_0"	Variable name
Address	"%M0"	Modbus address of the variable ("Addressing data from a PLC project" on page 162)
Connection	"BRC_Control"	Connection name
Data type	"Bit"	WinCC-flexible variable data type (fig. 5-55 "WinCC-flexible data types/IndraLogic data types" on page 166)
Array count	"1"	Number of array elements if variable is to be assigned as array
Acquisition cycle	"1s"	Update rate in which the variable values are to be read from the control

Fig. 5-57: WinCC - flexible variable declaration

In WinCC-flexible 2007, the following data types are available. The corresponding IndraLogic data types are listed below.

Application Samples

WinCC - flexible data types

WinCC - flexible	IndraLogic	Description
Int	UINT	Unsigned integer length 16 bit Value range: 0 to 65535
+/-int	INT	Signed integer length 16 bit Value range: -32768 to 32767
Double	UDINT	Unsigned double integer length 32 bit Value range: 0 to 4294967295
+/-double	DINT	Signed double integer length 32 bit Value range: -2147483648 to 2147483647
Float	REAL	Floating point length 32 bit Value range: 1.175494351e-38F to 3.402823466e+38F
Bit	Single bit of a byte	Single bit length 1 bit ("Bit data" on page 162)
16 bit group	WORD	WORD length 16 bits Attention: For 16 bit groups, the bit counting in WinCC-flexible is vice versa to the one in IndraLogic
ASCII	STRING	The string length (in bytes) is to be configured in WinCC-flexible via the Tag Properties. ("Text string" on page 163)

Fig. 5-58: WinCC-flexible data types/IndraLogic data types

The declared variables can now be used for different WinCC-flexible functionalities.

Example:

Displaying and entering different variables

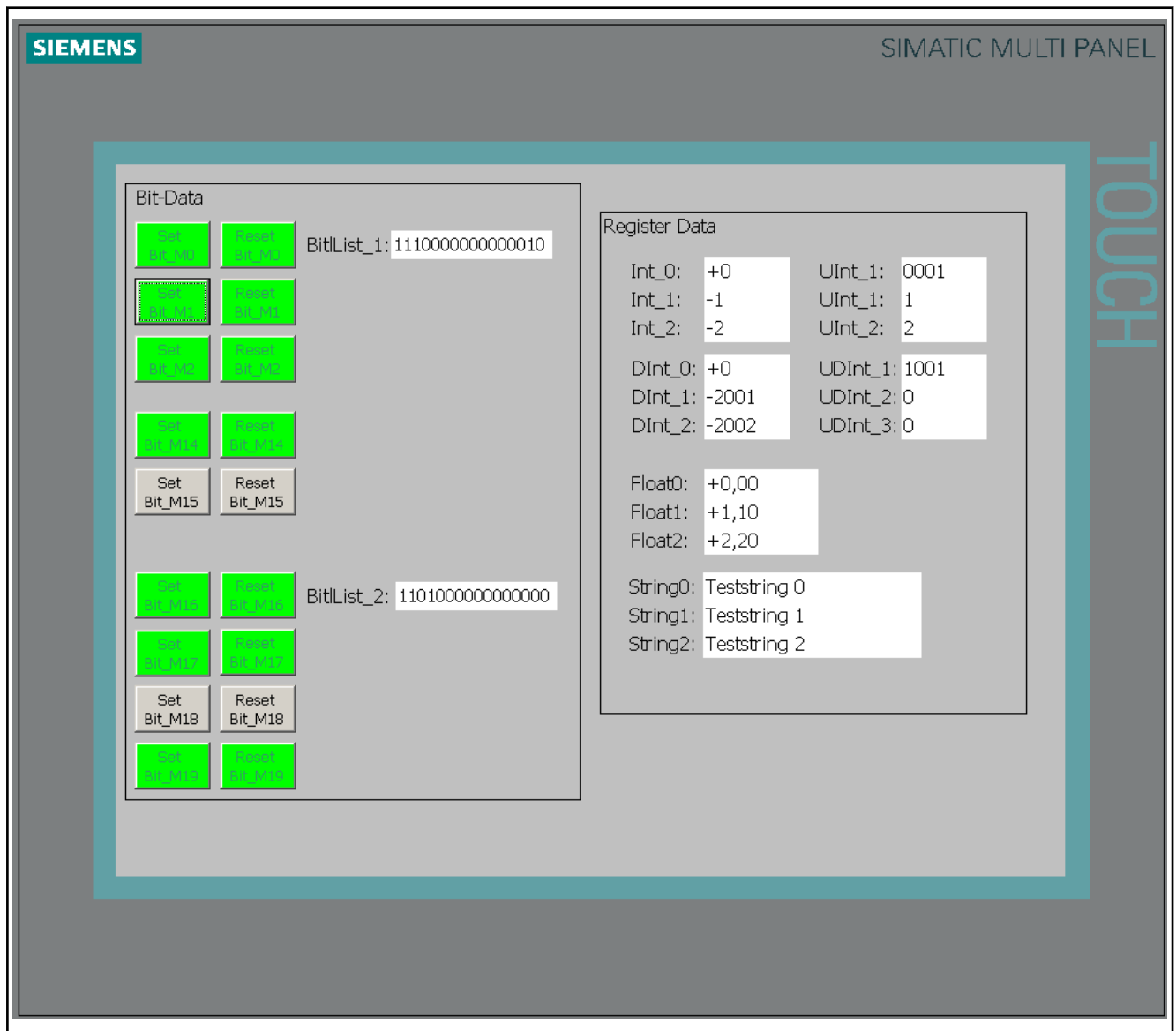


Fig.5-59: WinCC-flexible - Visualization example

Configuration with Pro-face GP-Pro EX 2.2

Communication settings

The following settings are required to configure the connection in Pro-face GP-Pro EX 2.2:

1. In the menu **System settings** select **"Device/PLC"**
2. Execute the **"Add Device/PLC"** link in the working area of the window
3. Subsequently, the following settings have to be selected in the "Add Device/PLC" dialog:

Application Samples

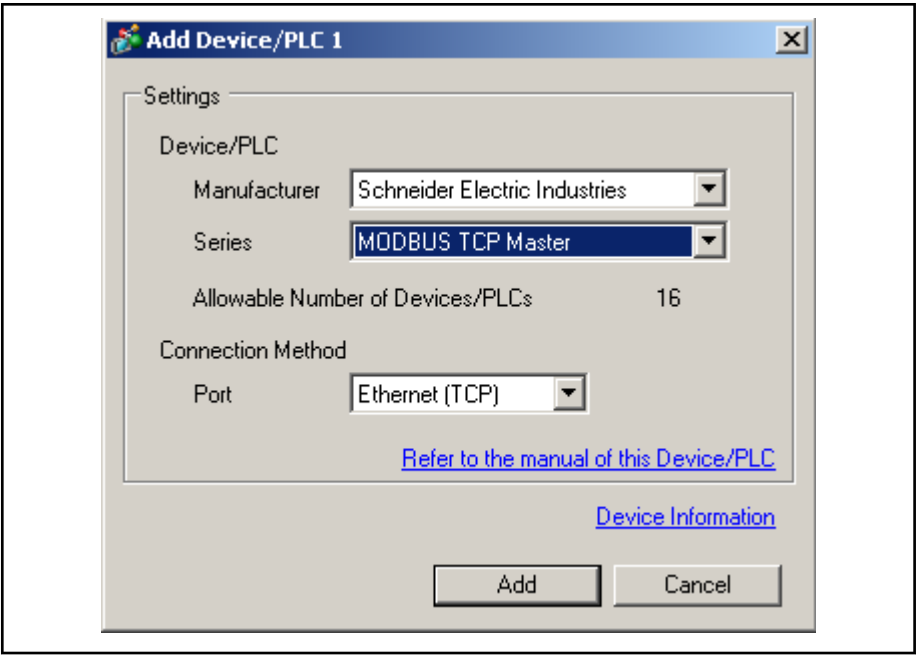


Fig.5-60: GP-Pro EX communication settings - Device selection

Drop-down menu	Selection
Manufacturer	Schneider Electric Industries
Series	MOCBUS TCP master
Port	Ethernet (TCP)

Fig.5-61: Communication settings - Device selection

Acknowledge your selection by clicking on **Add**

4. Enter the connection name in the extended window of the working area under "Device Name" and click on  **"Settings"**.

Device/PLC

[Add Device/PLC](#) [Delete Device/PLC](#)

Device/PLC 1

Summary [Change Device/PLC](#)

Manufacturer Series Port

Text Data Mode [Change](#)

Communication Settings

Timeout (sec)

Retry

Wait To Send (ms)

Device-Specific Settings

Allowable Number of Devices/PLCs 16

Number	Device Name	Settings
1	BRC_Control	Settings IP Address=000.000.000.000,Port No.=502,Unit ID=255,Rest of the bits in this

Fig.5-62: GP-Pro EX communication settings - Working range

- Subsequently, the following settings have to be selected in the "Equipment Configuration" tab in the "Individual Device Settings" dialog:

Application Samples

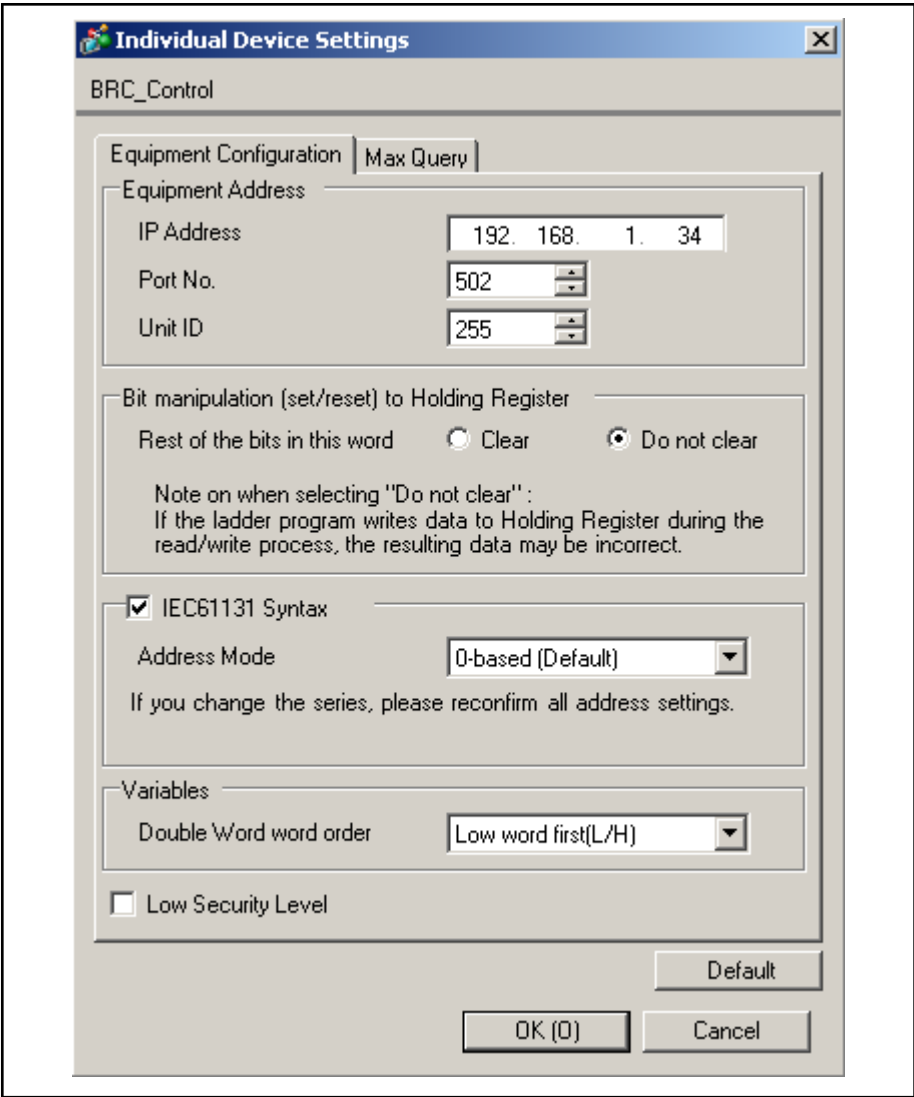


Fig.5-63: GP-Pro EX communication settings - Device parameters

Section	Name	Value	Description
Equipment Address	IP Address	"192.168.1.34"	IP address of the control
	Type	"502"	Port of the communication connection (port of the IL_ModbusTCPServer function block instance)
	Unit ID	"255"	Maintain default setting (255) ! not relevant for the Modbus TCP - refer to "MODBUS Messaging on TCP/IP Implementation Guide V1.0b" on www.Modbus-IDA.org
Bit manipulation to Holding Register	Reset of the bits in this word	"Do not clear"	Maintain default setting (Do not clear)

Application Samples

Section	Name	Value	Description
IEC61131 Syntax	Address Mode	"0-based (default)"	Activate IEC61131 Syntax and maintain Address Mode for the default setting (0-based)
Variables	Double Word word order	"Low word first (L/H)"	Word order within a double word

Fig. 5-64: GP-Pro EX communication settings - Device parameters

The "MaxQuery" register defines the maximum number of data that can be packed in the Modbus request. The default settings can be retained.

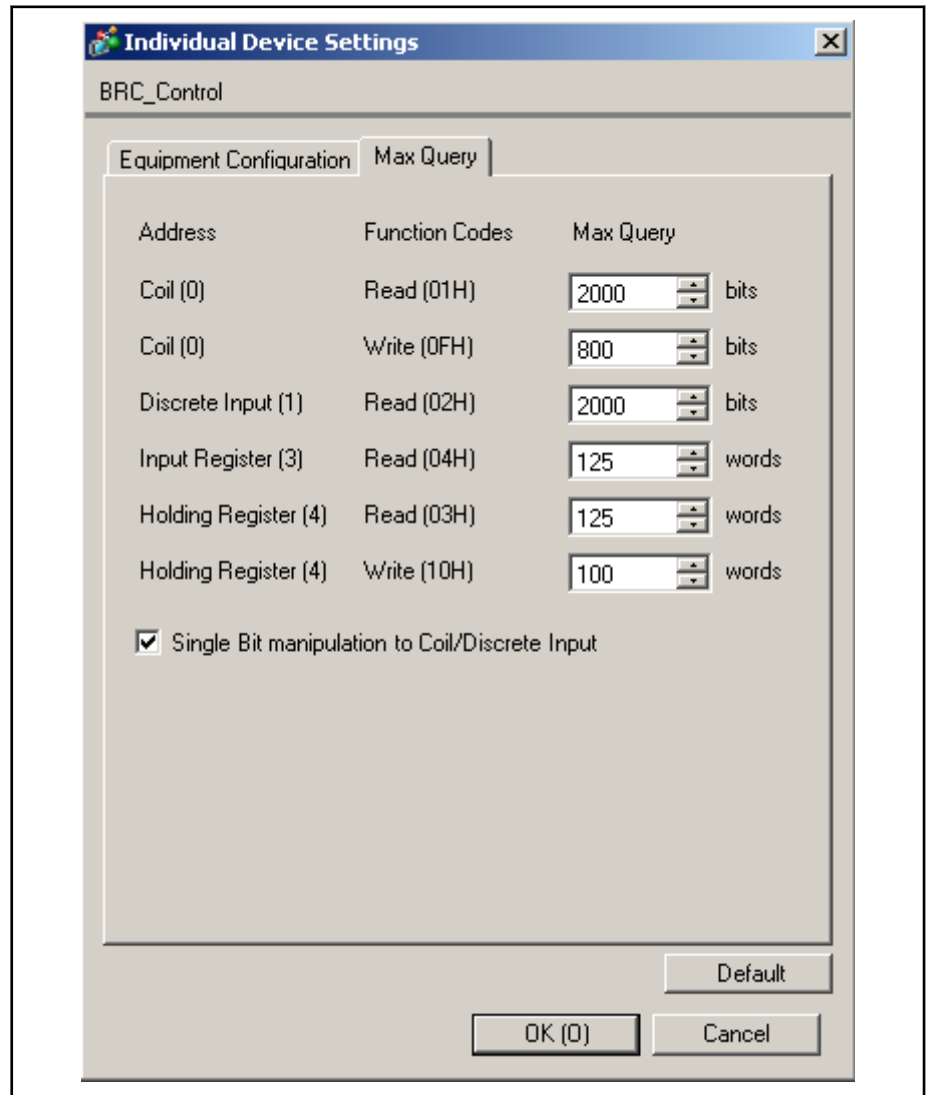
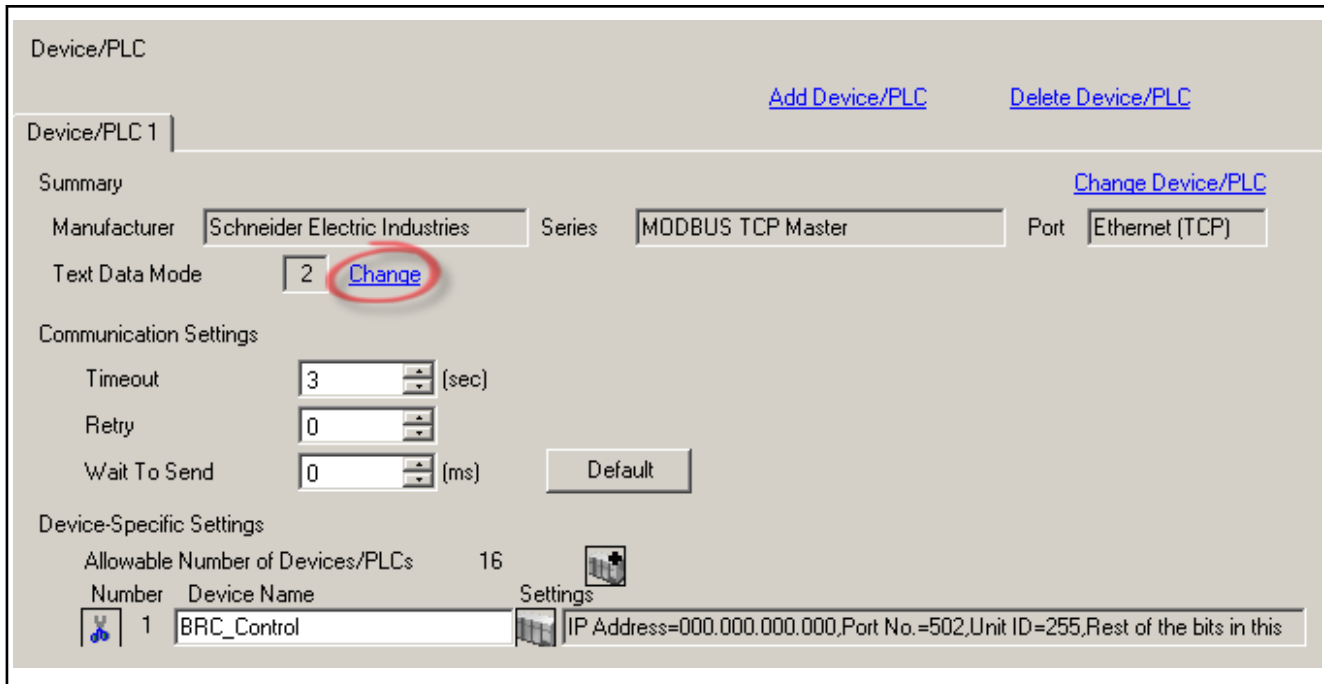


Fig. 5-65: GP-Pro EX communication settings - Device parameters

Confirm selection with <OK>

- The "Text Data Mode" have to be adjusted to allow interpretation of texts. Therefore, click on the **"Change"** link in the working area window.

Application Samples



Device/PLC

[Add Device/PLC](#) [Delete Device/PLC](#)

Device/PLC 1

Summary [Change Device/PLC](#)

Manufacturer Series Port

Text Data Mode [Change](#)

Communication Settings

Timeout (sec)

Retry

Wait To Send (ms)

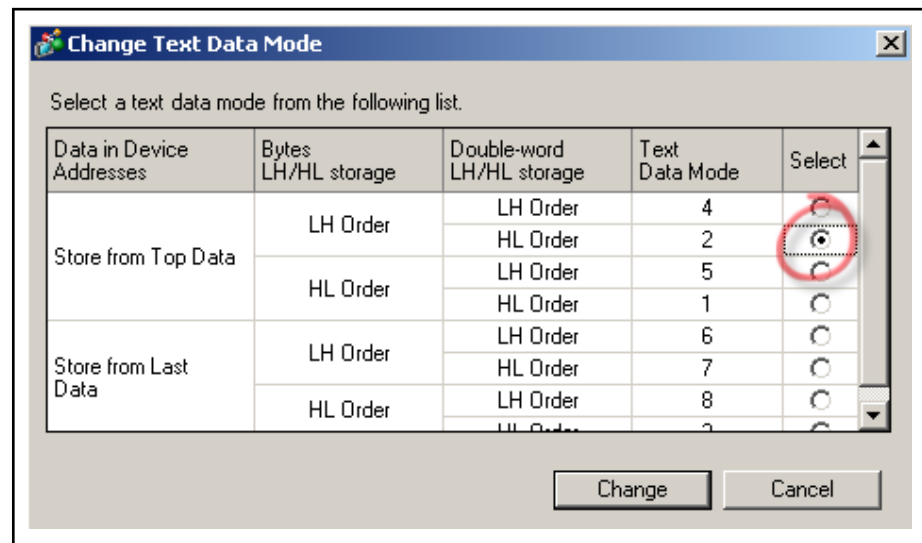
Device-Specific Settings

Allowable Number of Devices/PLCs 16

Number	Device Name	Settings
1	BRC_Control	IP Address=000.000.000.000,Port No.=502,Unit ID=255,Rest of the bits in this

Fig.5-66: GP-Pro EX communication settings - Working range

Subsequently, select 2 in the "Change Text Data Mode" dialog and confirm with the <Change> button.



Change Text Data Mode

Select a text data mode from the following list.

Data in Device Addresses	Bytes LH/HL storage	Double-word LH/HL storage	Text Data Mode	Select
Store from Top Data	LH Order	LH Order	4	☐
		HL Order	2	☒
	HL Order	LH Order	5	☐
		HL Order	1	☐
Store from Last Data	LH Order	LH Order	6	☐
		HL Order	7	☐
	HL Order	LH Order	8	☐
		HL Order	9	☐

Fig.5-67: Dialog: Change Text Data Mode (GP-Pro EX communication settings)

Using addresses

After configuring the connection, the addresses can be used directly in different GP-Pro EX functionalities, refer to "Addressing data from a PLC project" on page 162.

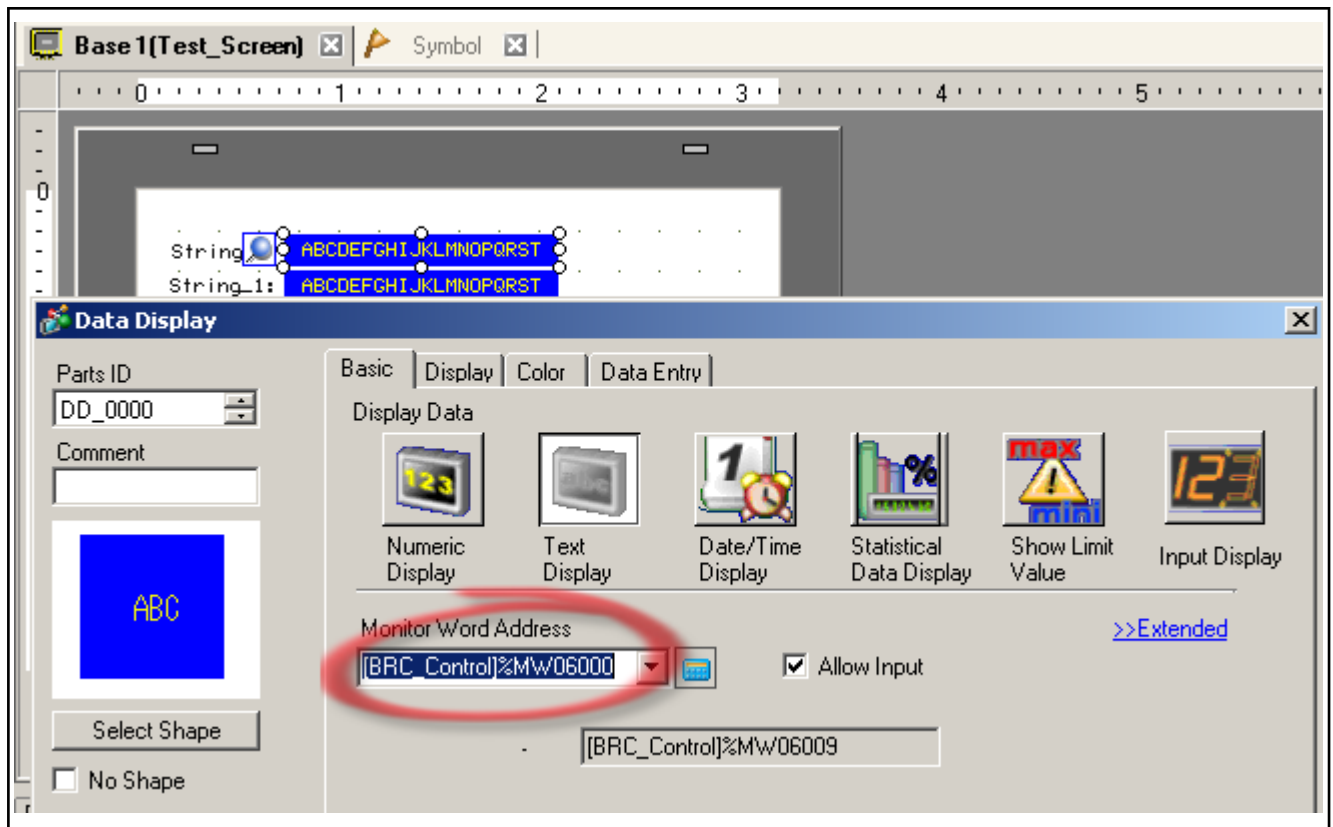


Fig.5-68: GP-Pro EX - Direct address use

Variable Declaration

Symbol variables of type bit address and word address can also be declared. For details, please refer to ["Addressing data from a PLC project" on page 162](#). The address content can be used in different GP-Pro EX functionalities via these symbol variables.

Via the menu **Common Settings** ► **Symbol Variables** in the working range, the window to declare symbol variables can be opened:

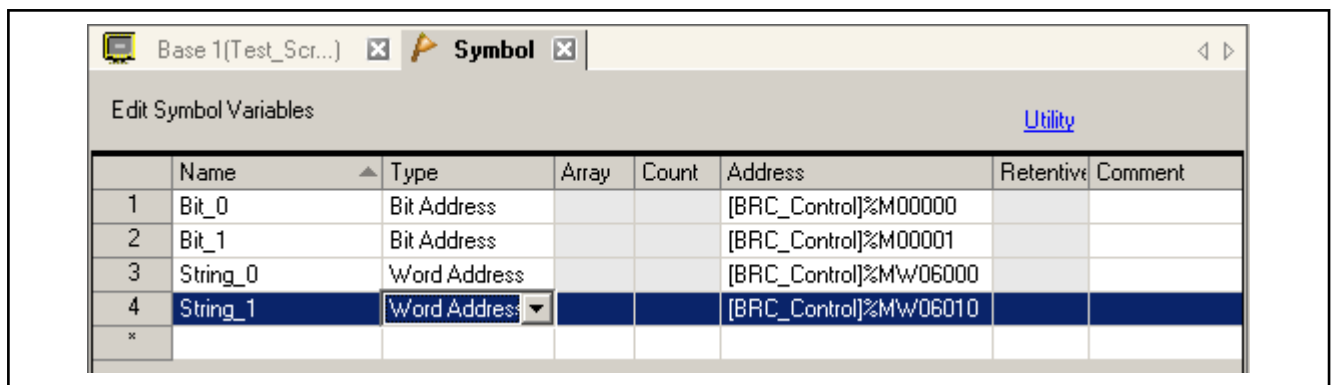


Fig.5-69: GP-Pro EX - Create a symbol variable

After creating the symbol variable, it can be used as "placeholder" for the address.

Application Samples

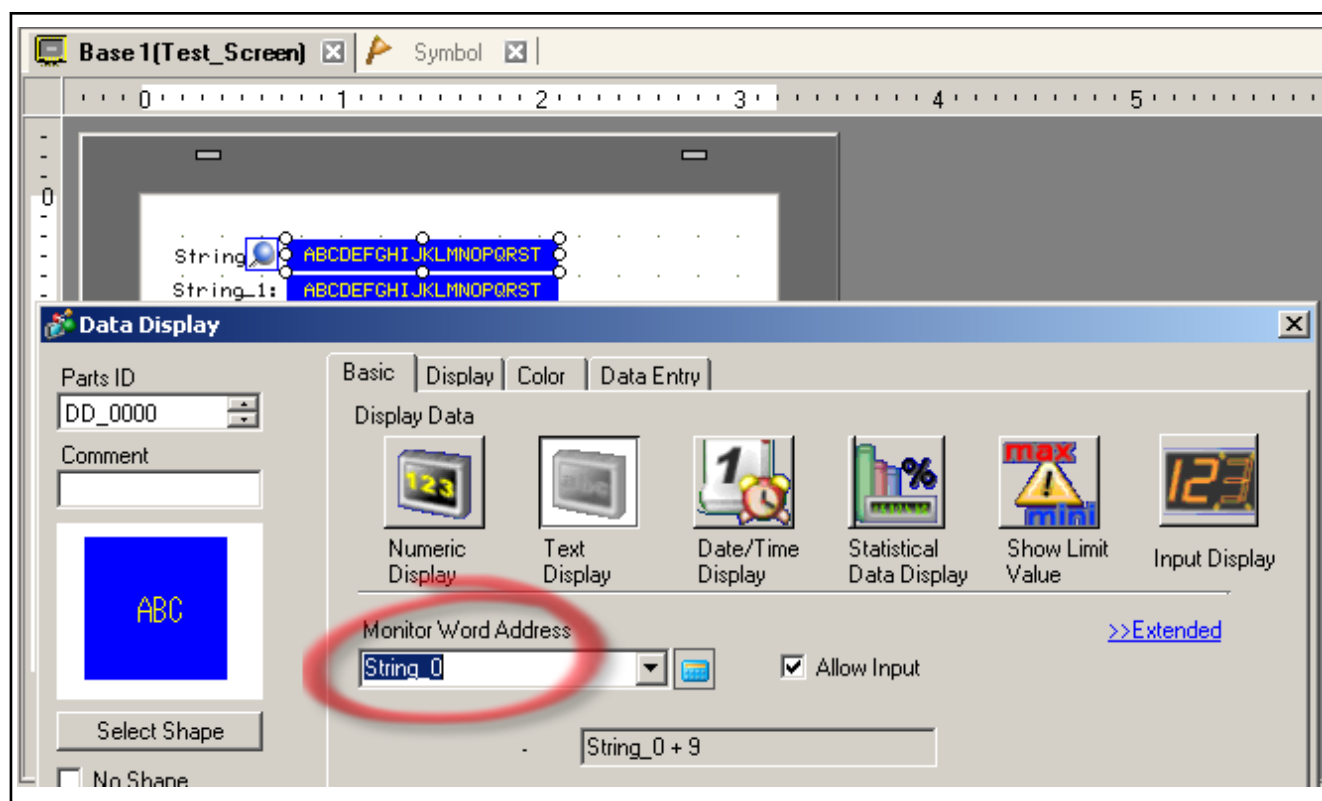


Fig.5-70: GP-Pro EX - Using symbol variables



The number of the registers to be interpreted is defined via the data type used in the individual display fields.

For display fields of type "Text Display", the number of characters to be displayed are defined via the number of the registers to be interpreted.

Example:

Displaying and entering different variables

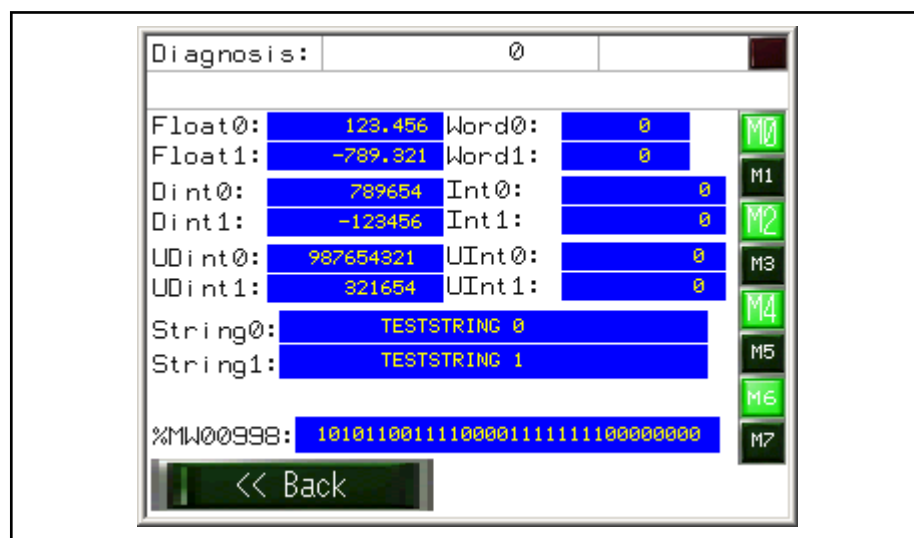


Fig.5-71: Variable display

6 Client Development

6.1 OPC Client Visual Basic .NET

6.1.1 General

This section deals with the development of the OPC Client "Visual Basic .NET" using "Microsoft Visual Studio .NET 2005".

This section deals with the following topics:

- Creating projects
- Adding References
- Adding Windows forms and control elements
- OPC Client implementation

The following diagram shows the OPC Client as an active Windows application.

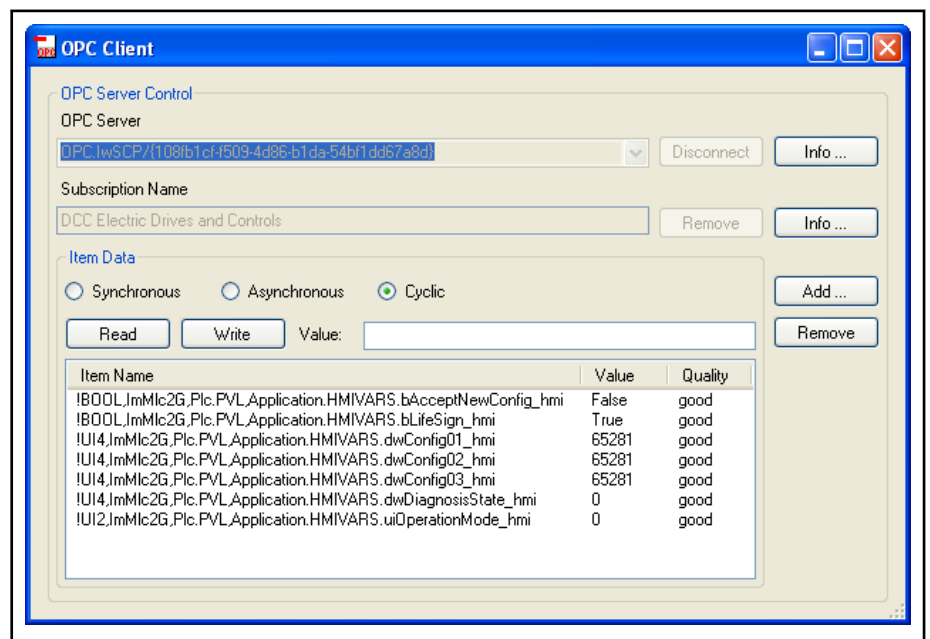


Fig. 6-1: OPC client interface

6.1.2 Visual Basic Project "OPC Client"

New Project

To create a new project in the "Microsoft Visual Studio .NET 2005" development environment, either click on the link "New Project" on the start page or in the menu **File** ► **New** click on the menu item **Project...**

The "New Project" dialog will appear, as shown below.

Project type and template can be selected here:

- "Other languages \ Visual Basic \ Windows" or
- "Visual Basic \ Windows" and
- "Windows application"

Enter "OPCCient" as the project name and adapt the suggested location accordingly.

The project is created by clicking on **OK**.

Client Development

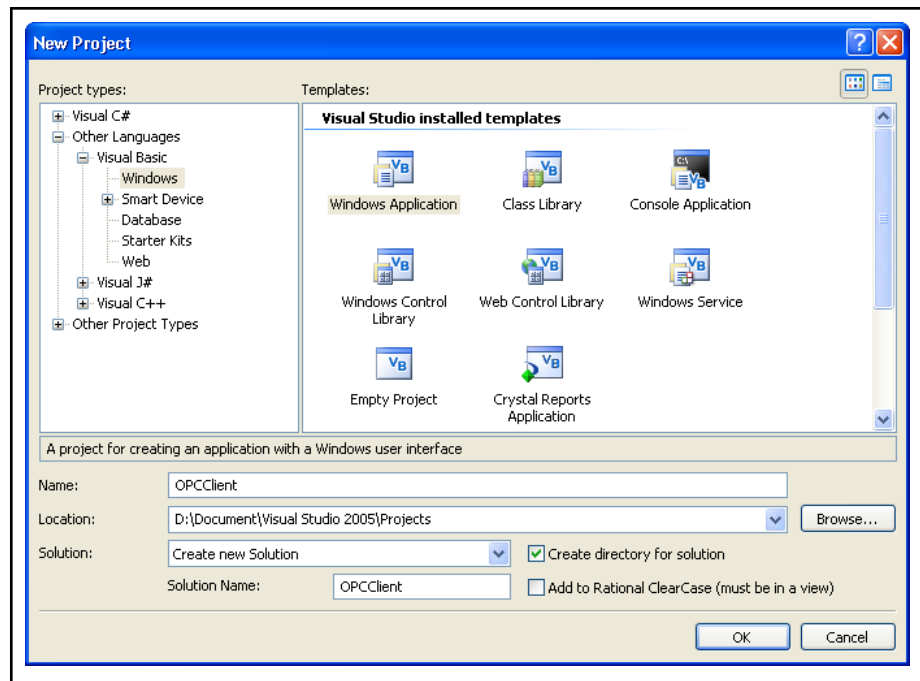


Fig.6-2: Dialog: New Project

Adding References

The project has to contain a reference to "OpcNetApi.dll" and "OpcNetApi.Com.dll" to allow access to the OPC .NET API functionalities.

"References" can be managed on the "Reference" page of the project properties. Additional dependencies can be added via the "Add Reference" dialog, which can be called up by clicking on "Add...". In the "Add Reference" dialog, the two files "OpcNetApi.dll" and "OpcNetApi.Com.dll" can be added to the project on the "Browse" tab.

Both files are provided with the "IndraWorks Suite" installation and are saved in the installation folder (e.g. "C:\Program Files\Rexroth\IndraWorks").

By opening the files they are carried over into the "References:".

The list "Imported namespaces:" shows all namespaces currently available for the project.

All selected namespaces can be used in the project without explicit declaration using the "Import" statement.

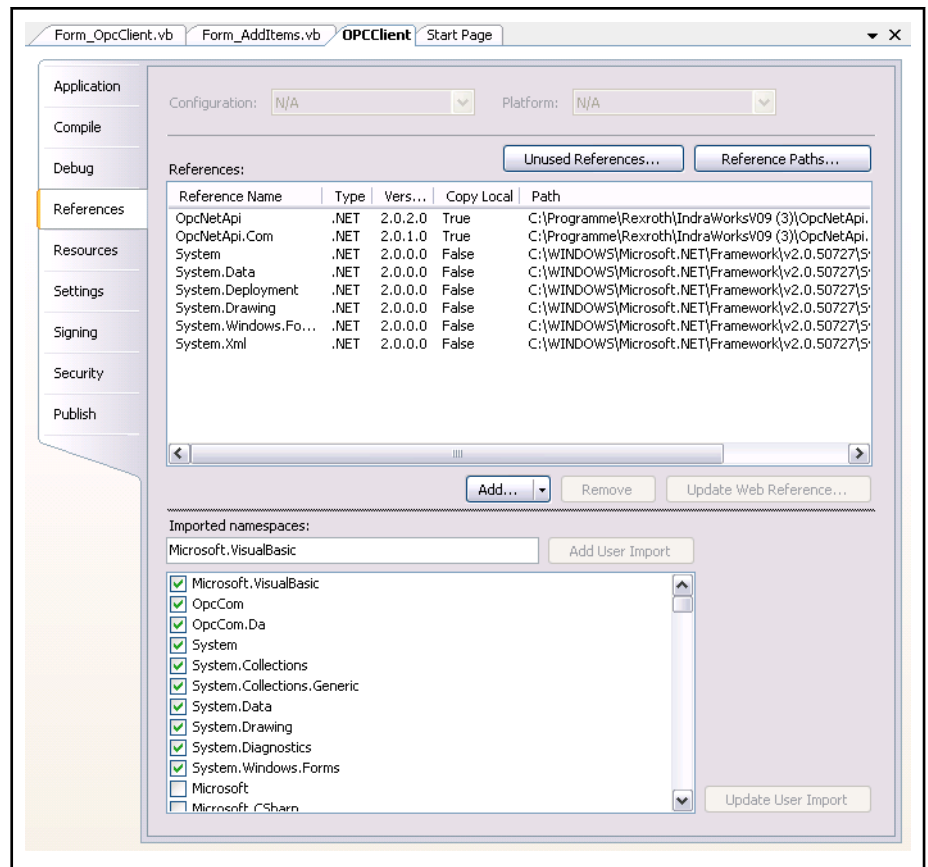


Fig. 6-3: Project properties - reference page

Adding Windows Form and Control Element

Add a windows form to the start project using the menu item **Project ► Add Windows Form....**

In the "Add New Element - OPCClient" dialog, the "Windows form" template has already been selected and only the name of the form needs to be entered.

All Windows forms created in the OPC client example have the name of the identifier "Form_" as a prefix.

The following figure shows an overview of the "OPC Client" project in the "Solution Explorer".

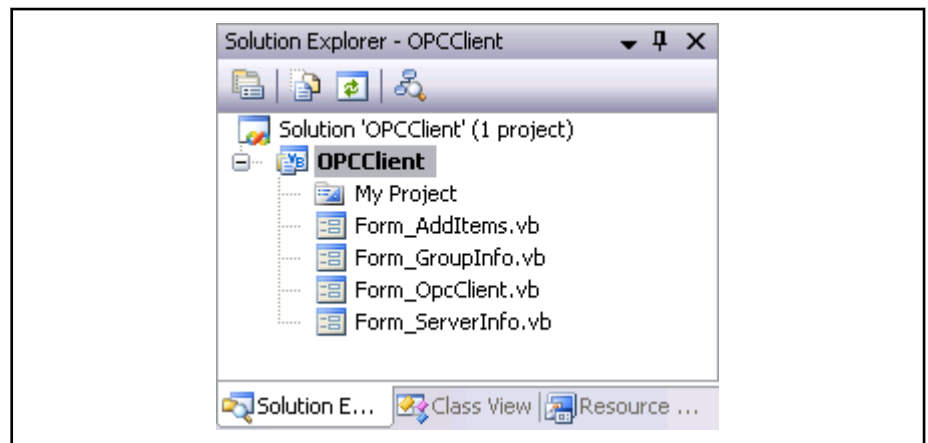


Fig. 6-4: Project overview

Client Development

"Windows forms" are designed with the "Windows Forms Designer" and the control elements available in the "Toolbox". Select the control elements and place them on the form.

The control elements for the selected object are configured in the "Properties" window.

Size and position of the control elements can also be changed using the mouse in the Windows Forms Designer.

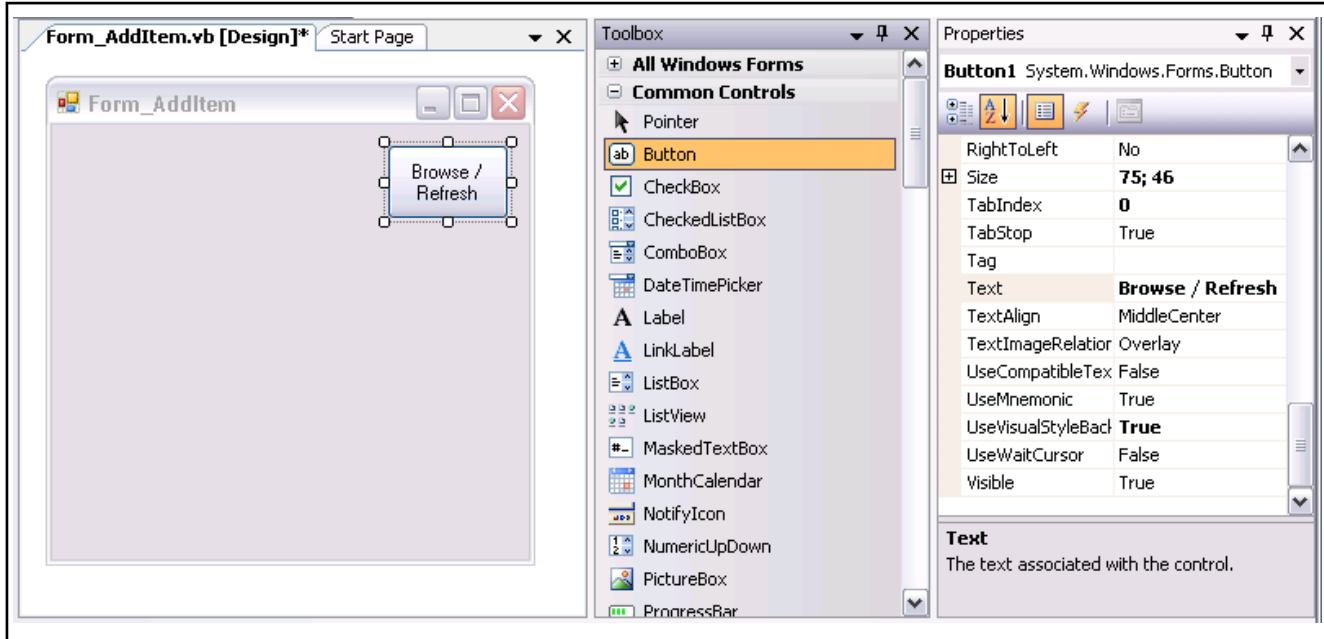


Fig.6-5: Adding and configuring control elements

The following figure shows the "OPC Client" form in "Windows Forms Designer".

To maintain clarity we will not provide a detailed description of the form and the individual control elements. All configurable properties can be taken from the example project.

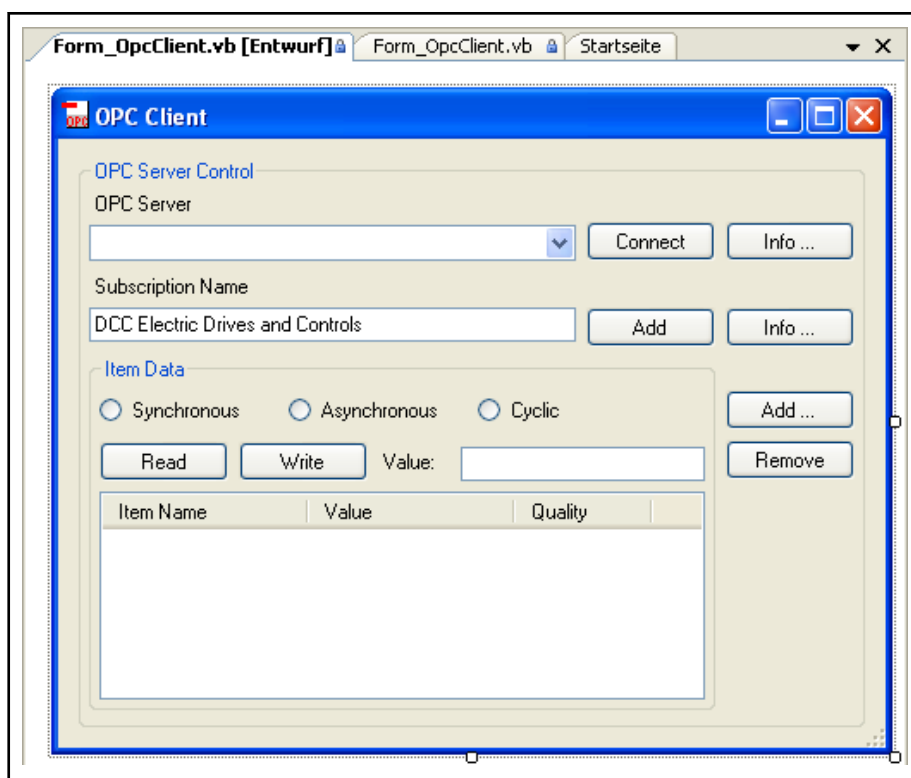


Fig.6-6: Dialog: OPC Client (Windows Forms Designer)

Implementing OPC Client

General

The following section describes the procedure for programming the OPC client.

To provide better clarity in the program, individual regions of the class have been combined with the "#Region" directive. These areas can be expanded or collapsed when creating the program.

The figure below shows the defined regions using the example of the "Form_OpcClient" form.

Range	Description
"#Region" -"#End Region"	
"Declaration"	Contains declaration of the class (member variables and constants).
"Form Events"	Procedures for handling the events triggered by the form and the control elements contained in it.
"OPC Events"	Procedures for handling the events triggered by the OPC server.
"Additional Internal Functions"	Additional internal definitions, structures and functions for optimizing user guidance.

Fig.6-7: Program structuring using "#Region" directives

The relevant program lines in the example project are also identified with the keyword "Step:" and numbered consecutively. This means that by using the search "Search in Files" (<Ctrl>+<Shift>+<F>) contained in Visual Studio al-

Client Development

lows branching directly to the implementation described by entering the search word e.g. **"Step: 2"**.

The implementation described in the following has been documented following the example of the operating order of the "OPC client" example application.

The structure is shown in the table below:

Chapter title	Description
chapter "Declaring Variables" on page 180	Declaration of the variables for managing OPC server and the Windows form.
chapter "Listing OPC server" on page 181	Listing of all OPC servers installed on the local PC.
chapter "Creating and Canceling OPC Server Instances" on page 182	Creating and canceling the OPC server instance and displaying the OPC server information.
chapter "Creating and canceling OPC subscription" on page 183	Creating and canceling an OPC subscription and displaying the subscription information.
chapter "Listing Control Items" on page 186	Reading and displaying all control items (browsing)
chapter "Adding and Deleting OPC Items" on page 187	Adding and deleting OPC items.
chapter "Reading Control Data" on page 190	Reading "synchronous" and "asynchronous" control data.
chapter "Writing control data" on page 193	Writing "synchronous" and "asynchronous" control data.
chapter "Cyclic Reading of Control Data" on page 195	Cyclic Reading of Control Data

Fig.6-8: Overview of implementation structure

Declaring Variables

In the **"Step: 1"** declaration section the "Form_OpcClient" variables are declared. The chosen syntax for the names provides writing of all class variables with the prefix "m_" ("member variables") and "capitalizes" the constant definitions.

The variables are

- the object for the OPC server instance and
- variables for managing the form.

The line **"Dim m_OpcServer As Opc.Da.Server = Nothing"** is the declaration relevant to the example. The "m_OpcServer" variable contains the created OPC server object.

```

' Step: 1
'
' declaration of members used for opc administration
#Region " Declaration "
'
' Step: 1
'
' declaration of members used for opc server administration
Dim m_OpcServer As Opc.Da.Server = Nothing

'
' members to store configuration
Dim m_DialogAddItems As OPCClient.Form_AddItems
Dim m_InternalStateControl As InternalStateControl

'
' constant values
Const BUTTON_DESCRIPTION_SERVER_CONNECT As System.String = "Connect"
Const BUTTON_DESCRIPTION_SERVER_DISCONNECT As System.String = "Disconnect"
Const BUTTON_DESCRIPTION_GROUP_ADD As System.String = "Add"
Const BUTTON_DESCRIPTION_GROUP_REMOVE As System.String = "Remove"
Const MAX_ITEMS As System.Int32 = 10

#End Region

```

Fig.6-9: Declaration of class variables

Listing OPC server

All OPC servers installed locally on the PC are listed in the "Load Event" "Step: 2" of the "Form_OpcClient" form. All OPC servers are requested via the "opcDiscovery.GetAvailableServers()" method of the Discovery interface "Opc.IDiscovery" of the "OpcEnum.exe" service, which support the specification "Data Access 2.0". The available OPC specifications are defined in the structure "Opc.Specification".

The list provided by the Discovery interface contains the complete URL path with ProgID (without version index) and GUID to clearly identify the server.

```

' Step: 2
'
' browse for available opc servers on this system
Private Sub FormOpcClient_Load(ByVal sender As Object, ByVal e As System.EventArgs)

    Dim opcServer As Opc.Server
    Dim opcServers As Opc.Server()
    Dim opcDiscovery As Opc.IDiscovery

    Try

        ' Step: 2
        '
        ' browse for available opc servers on this system
        opcDiscovery = New OpcCom.ServerEnumerator
        opcServers = opcDiscovery.GetAvailableServers( _
            Opc.Specification.COM_DA_20)

        ' copy opc server to ...
        Me.ComboBox_Server.BeginUpdate()

        For Each opcServer In opcServers
            ' add all server to the combo box
            Me.ComboBox_Server.Items.Add(opcServer.Url.Path)
        Next

    Catch ex As Exception
    End Try
End Sub

```

Fig.6-10: Listing installed OPC servers

Creating and Canceling OPC Server Instances

General

This section describes how to create and cancel the OPC server instance and how to read the server information. The OPC Server instance is managed in the event handler "Button_Connect_Disconnect_Click" of the button "Connect" or "Disconnect", reading and displaying the server information is carried out in the function "Button_ServerInfor_Click" of the "Info ..." button.

The relevant code segments in the OPC client example project are:

- "Step: 3" - create OPC server instance
- "Step: 4" - delete OPC server instance
- "Step: 5" and "Step: 21" - read server information

Creating OPC server instance - Connect

The creation of an OPC server instance and the connection establishment is described in "Step: 3" (see the following figure).

The first step creates an OPC server instance "New Opc.Da.Server" with transfer of the "OpcCom.Factory" parameter and the "Opc.URL".

The "Opc.URL" contains information about the OPC specification "Opc.UrlScheme.DA", the PC name "localhost" or the IP address on which the server is to be started, as well as the server identifier.

In the example project, the server identifier is read from the combination list field "ComboBox_Server".

The following options are possible for specifying the server identifier:

- "OPC.lwSCP"
ProgID without version index
- "OPC.lwSCP.1"
ProgID with version index
- "OPC.lwSCP/{108fb1cf-f509-4d86-b1da-54bf1dd67a8d}"
ProgID without version index and GUID
- "OPC.lwSCP.1/{108fb1cf-f509-4d86-b1da-54bf1dd67a8d}"
ProgID with version index and GUID

The OPC server is instanced by calling the function "Me.m_OpcServer.Connect()".

Deleting OPC server instance - Disconnect

Deletion of the OPC server instance is shown in "Step: 4".

Calling the function "Me.m_OpcServer.Disconnect()" causes log-off from the OPC server. If no further client is connected to the server at this point, it will be terminated.

The function "Me.m_OpcServer.Dispose()" will then be called to release resources which are held in the OPC server instance and the instance is canceled by "Me.m_OpcServer = Nothing".

```

' show hour glas
Me.Cursor = Cursors.WaitCursor

If Me.Button_Connect_Disconnect.Text = BUTTON_DESCRIPTION_SERVER_CONN

    ' Step: 3
    '
    ' select opc server and
    Me.m_OpcServer = New Opc.Da.Server(New OpcCom.Factory, _
    New Opc.URL(Opc.UriScheme.DA + "://localhost/" + _
    Me.ComboBox_Server.Text))

    ' connect with ...
    Me.m_OpcServer.Connect()

    ' Enable or disable controls
    Me.ComboBox_Server.Focus()
    Me.SetInternalStateControl(InternalStateControl.Connected)

ElseIf Me.Button_Connect_Disconnect.Text = BUTTON_DESCRIPTION_SERVER_

    ' Step: 4
    '
    ' disconnect from server and clean up
    Me.m_OpcServer.Disconnect()
    Me.m_OpcServer.Dispose()
    Me.m_OpcServer = Nothing

    ' Enable or disable controls
    Me.SetInternalStateControl(InternalStateControl.Disconnected)

    ' clear browse info
    If Not Me.m_DialogAddItems Is Nothing Then Me.m_DialogAddItems.Cle

End If

```

Fig.6-11: Creating and canceling OPC server instance

Reading server information

The information provided by the OPC Server is read and displayed in the "Load Event" "Step: 21" of the form "Form_ServerInfo".

The form is called in "Step: 5" (no figure).

```

' Step: 21
'
' read opc server information
Private Sub Form_ServerInfo_Load(ByVal sender As System.Object, ByVal e As :

    Try

        '
        If (Me.m_OPcServer Is Nothing) Then Exit Sub

        ' Step: 21
        '
        ' read opc server information
        Me.Label_Name.Text = m_OPcServer.Name
        Me.Label_URL.Text = m_OPcServer.Url.Path
        Me.Label_Status.Text = m_OPcServer.GetStatus.StatusInfo
        Me.Label_Vendor.Text = m_OPcServer.GetStatus.VendorInfo

    Catch ex As Exception

```

Fig.6-12: Reading OPC server information

Creating and canceling OPC subscription

General

This section describes login and logout of a "subscription" (group) on the OPC server. The subscription is managed in the "Button_Add_Re-

Client Development

move_Group_Click" event handler of the **"Add"** or **"Remove"** button and the subscription information is read and displayed in the "Button_GroupInfo_Click" function of the **"Info ..."** button.

The relevant code segments in the OPC client example project are:

- **"Step: 6"** - create subscription
- **"Step: 7"** - cancel subscription
- **"Step: 8"** and **"Step: 22"** - display subscription information

Creating a subscription - CreateSubscription

A subscription is created in the OPC server collection with the function "Me.m_OpcServer.CreateSubscription()" and transfer of the parameter "Opc.Da.SubscriptionState" (**"Step: 6"**, see [fig. 6-14 "Creating an OPC subscription" on page 185](#)).

The elements of the "Opc.Da.SubscriptionState" class serve to clearly identify the subscription and to configure it in relation to its behavior or processing by the OPC server.

The following table provides a description of the class elements.

Class element	Description
Active	The control cyclically reads the subscription items and saves them in the server cache.
ClientHandle	"Unique Identifier" (issued by the client).
Deadband	Smallest possible percentage change of a value to trigger an event to the client.
KeepAlive	Largest possible time interval between two events to the client (life sign).
Locale	Definition of localization in the format "[language Code]-[country/region code]", e.g. "de-DE", "en-US"
Name	"Unique Name" (issued by the client). Specified by the "TextBox_Group.Text" text box.
ServerHandle	"Unique Identifier" (issued by the server).
UpdateRate	Time interval during which the server checks the data and sends an event to the client.

Fig. 6-13: Elements of class "Opc.Da.SubscriptionState"

```

' Step: 6 / 7
'
' create and cancel subscription
Private Sub Button_Add_Remove_Group_Click(ByVal sender As System.Object, By

Dim subSubscriptionState As Opc.Da.SubscriptionState
Dim iSubscription As Opc.Da.ISubscription

Try

    ' show hour glas
    Me.Cursor = Cursors.WaitCursor

    If Me.Button_Add_Remove_Group.Text = BUTTON_DESCRIPTION_GROUP_ADD Then

        ' Step: 6
        '
        ' create subscription
        subSubscriptionState = New Opc.Da.SubscriptionState
        subSubscriptionState.Active = True ' activate cyclic communication
        subSubscriptionState.ClientHandle = System.Guid.NewGuid().ToString()
        subSubscriptionState.Deadband = 0.0F
        subSubscriptionState.Name = TextBox_Group.Text
        subSubscriptionState.UpdateRate = 1000

        Me.m_OpcServer.CreateSubscription(subSubscriptionState)

        ' Enable or disable controls
        Me.SetInternalStateControl(InternalStateControl.SubscriptionSet)

    ElseIf Me.Button_Add_Remove_Group.Text = BUTTON_DESCRIPTION_GROUP_REM

        ' Step: 7

```

Fig.6-14: Creating an OPC subscription

Canceling a subscription - CancelSubscription

The function "Me.m_OpcServer.CancelSubscription()" is requested with the transfer of the parameter "Opc.Da.ISubscription" to cancel a subscription ("Step: 7").

Once the subscription has been canceled from the OPC server collection it is essential that the "iSubscription.Dispose()" function is called so that the resources used by the subscription can be released again. The reference of the interface variables is canceled by the assignment "iSubscription = Nothing".

The local assignment "iSubscription = Me.m_OpcServer.Subscription()" performed at the start of the code segment is necessary, as the subscription was removed from the OPC server collection following calling of function "Me.m_OpcServer.CancelSubscription()" and is no longer available.

```

ElseIf Me.Button_Add_Remove_Group.Text = BUTTON_DESCRIPTION_GROUP_REM

    ' Step: 7
    '
    ' cancel subscription and clean up
    iSubscription = Me.m_OpcServer.Subscriptions(0)
    Me.m_OpcServer.CancelSubscription(iSubscription)
    iSubscription.Dispose()
    iSubscription = Nothing

    ' Enable or disable controls
    Me.SetInternalStateControl(InternalStateControl.Connected)

End If

Catch ex As Exception

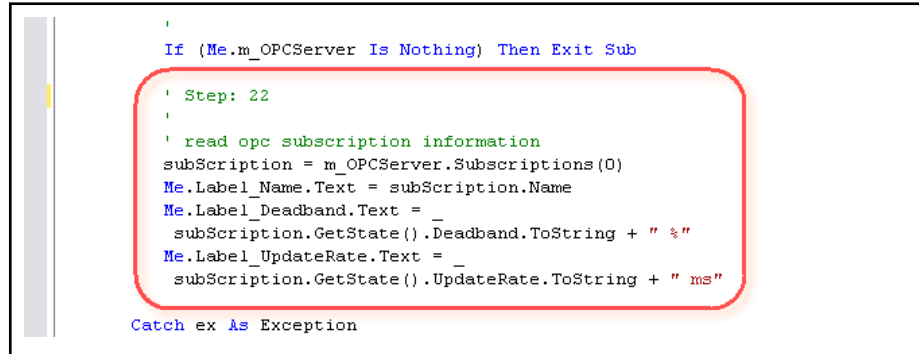
```

Fig.6-15: Canceling OPC subscription

Displaying subscription information

The information provided by the subscription is read and displayed in the "Load Event" ("**Step: 22**") in the form "Form_GroupInfo".

The form is called in "**Step: 8**" (no figure).



```

'
If (Me.m_OPCTServer Is Nothing) Then Exit Sub

' Step: 22
'
' read opc subscription information
subScripton = m_OPCTServer.Subscriptions(0)
Me.Label_Name.Text = subScripton.Name
Me.Label_Deadband.Text = _
subScripton.GetState().Deadband.ToString + " %"
Me.Label_UpdateRate.Text = _
subScripton.GetState().UpdateRate.ToString + " ms"

Catch ex As Exception

```

Fig.6-16: Reading OPC subscription information

Listing Control Items

General

This section describes the reading of all items configured in the control for access via the OPC server.

The relevant code segments in the OPC client example project are:

- "**Step: 20**", "**Step: 20.1**" and "**Step: 20.2**" - reading control items

Reading control items

Clicking on the "Add ..." button on the "Form_OpcClient" form calls the "Form_AddItems" form. As it can take some time to read the items, the items available on the OPC server are not read in the "Load Event" of the form but by using the "Button_BrowseItems_Click" function which is activated by clicking on the "**Browse / Refresh**" button.

The two classes "Opc.ItemIdentifier" and "Opc.Da.BrowseFilters" of the "Me.m_OPCTServer.Browse()" function are transferred as parameters to read the items ("**Step: 20.2**").

The classes have the following function:

Opc.ItemIdentifier	The class "Opc.ItemIdentifier" sets the "starting point" for the search in the "tree hierarchy". Consequently, when e.g. a search was canceled the search can recommence at this point.
Opc.Da.BrowseFilter	The class "Opc.Da.BrowseFilter" serves as a filter criterion for the result of the function call. There is an option here to only read variables, which begin e.g. with a certain sequence of characters.

As there is no default specification for the format in which an OPC server returns the "browsing" information, reading of the control items was divided into two functions ("**Step: 20.1**" and "**Step: 20.2**"). This division allows reading of flat hierarchies (tree structure without sub-nodes) and structured hierarchies (tree structure with sub-nodes) (see following figures).

```

' clear list, stop updating
Me.ListBox_Available_Items.BeginUpdate()
Me.ListBox_Available_Items.Items.Clear()

' Step: 20.1
'
' browse items
filters.ElementNameFilter = ""
filters.BrowseFilter = Opc.Da.browseFilter.all
filters.ReturnAllProperties = True
filters.ReturnPropertyValues = True
itemID.ItemName = ""
Me.BrowseItems(itemID, filters)

Catch ex As Exception

```

Fig.6-17: Reading control items - filter configuration

```

' Step: 20.2
'
' browse items
elements = m_OPCTServer.Browse(ItemID, Filters, position)

' adds each element to the list
For Each element In elements
    ' avoid errors while using .NET API 1.30
    If Not element.ItemName Is Nothing Then
        itemName = element.ItemName
    Else
        itemName = element.Name
    End If
    Me.ListBox_Available_Items.Items.Add(itemName)

    If (element.HasChildren = True) Then
        ' check on subordinate items

        ' set branch to search
        item.ItemName = itemName & element.ItemName
        item.ItemPath = element.ItemPath
        Me.BrowseItems(item, Filters)

    End If
Next

```

Fig.6-18: Reading control items - recursion

Adding and Deleting OPC Items

General

This section describes the adding and deleting of items to/from a subscription.

The relevant code segments in the OPC client example project are:

- "Step: 9", "Step: 9.1" and "Step: 9.2" - adding items
- "Step: 10" - deleting items

Adding Items - AddItems

In the form event "Button_Add_Item_Click" ("Step: 9") the form "Form_AddItems" is called and the items selected in the form are added.

Client Development

The variables can be entered separately in the text field of the "Add Items" form by ";" or they can be selected from the list. When entering the variables, the item syntax of the respective OPC server is to be taken into account.

The following shows the item syntax for the "OPC.lwSCP" server and the "IndraLogic OPC" server.

"OPC.lwSCP" Server

- **PLC variables**

"![required data type],[control name or alias],[lwSCP service],[fully qualifying variable name]"

Example:

"!Bool,ImMlc2G,Plc.PVL,Application.HMIVARS.bLifeSign_hmi"

- **sercos parameter**

"![required data type],[control name or alias],[lwSCP service],[drive address], [parameter number], [element coding]"

Example:

"!UI4,ImMlc2G,SPA1,0,C-0-0001,D"

"!UI4,ImMlc2G,SPA1,0,C-0-0001,40"



The "OPC.lwSCP" server allows access to parameter and PLC variables!



Further information on the item syntax of the "OPC.lwSCP" server can be found in [chapter "lwSCP and OPC.lwSCP Items" on page 89](#).

"IndraLogic OPC" server

- **PLC variables**

"[Control name].[fully qualifying variable name]"

Example:

"PLC_IL2G.Application.HMIVARS.bLifeSign_hmi"



The "IndraLogic OPC" server allows access to PLC variables.

The description of the "Browse" functionality is described in [chapter "Listing Control Items" on page 186](#).

Once the items have been entered or selected in the "Add Items" form, in **"Step: 9.1"** an "Opc.Da.Item" class is created and initialized for every item. The name of the item is assigned in the line "opcltems(index).ItemName = items(index)".

The class elements of class "Opc.Da.Item" used in the example are shown in the following table:

Class element	Description
Active	"DataChanged" events for the item are send to the client.
ActiveSpecified	The "Active" property was defined.
ItemName	Fully qualified item name. The syntax is dependent on the OPC server.
ItemPath	Fully qualified item path. This property is not evaluated by the "OPC.lwSCP".

Class element	Description
MaxAge	When the item is read from the OPC server cache the value may not be older than defined in "MaxAge".
MaxAgeSpecified	The "MaxAge" property was defined. If this property is set to "True" the value is read from the OPC server "cache" or from the "device" as a function of "MaxAge". If the property is set to "False" the value is always read from the "device".

Fig.6-19: Elements of class "Opc.Da.Item"

Once all OPC items have been created the OPC item array is logged in to the subscription on the OPC server with the function request "Me.m_OpcServer.Subscriptions(0).AddItems()" ("Step: 9.2").

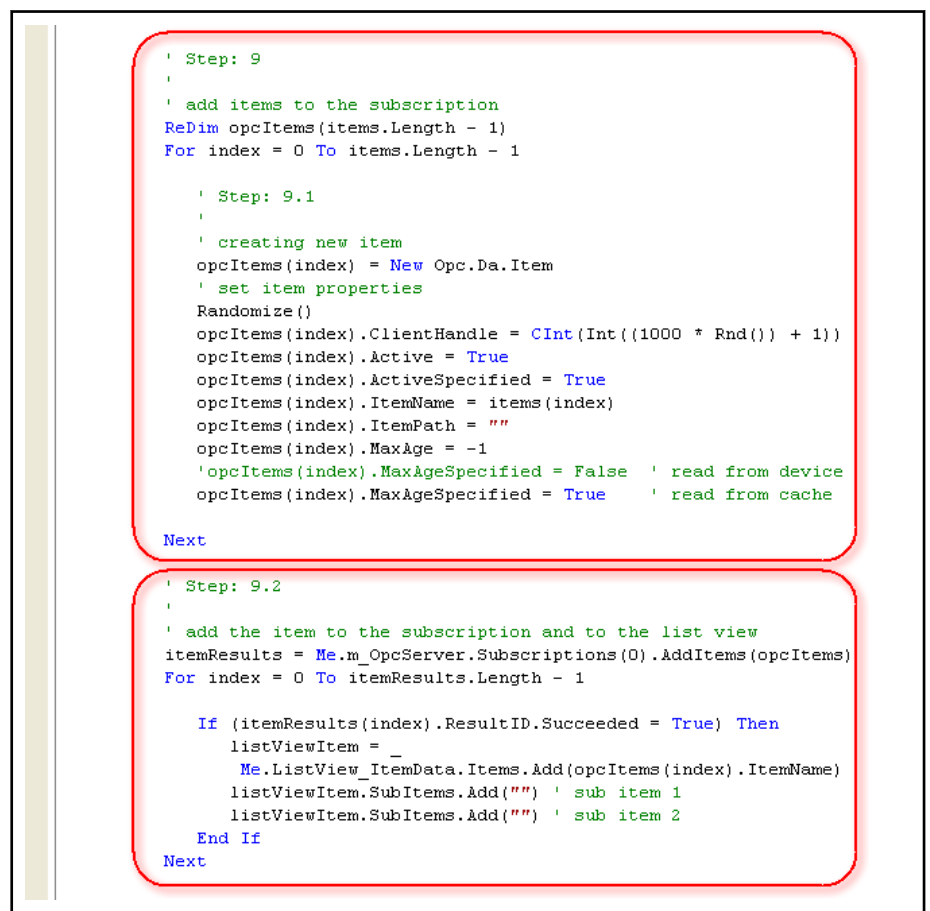


Fig.6-20: Adding items to a subscription

Removing Items - RemoveItems

To remove items from the OPC subscription, select the items from the list and click on "Remove". In the subsequently triggered "Button_Remove_Item_Click" form event the function "Me.m_OpcServer.Subscriptions(0).RemoveItems()" is called, which removes the OPC item transferred as a parameter ("Step: 10").

Client Development

```

Me.RadioButton_Sync.Checked = True

' set array size for opc item list
selectedItems = Me.ListView_ItemData.SelectedItems.Count - 1
ReDim opcItemIds(selectedItems)

' Step: 10
'
' remove selected items from opc subscription and from the list
For index = 0 To selectedItems
    For Each opcItem In Me.m_OpcServer.Subscriptions(0).Items
        If Me.ListView_ItemData.SelectedItems(index).Text.Equals(opcItem.ItemName) Then
            opcItemIds(index) = opcItem
            Exit For
        End If
    Next
Next
' remove items
opcIdentResult = Me.m_OpcServer.Subscriptions(0).RemoveItems(opcItemIds)

' clean up list view control
For index = 0 To opcIdentResult.Length - 1
    If (opcIdentResult(index).ResultID = ResultID.S_OK) Then
        For Each listViewItem In Me.ListView_ItemData.SelectedItems
            If (opcIdentResult(index).ItemName.Equals(listViewItem.Text)) Then
                Me.ListView_ItemData.Items.Remove(listViewItem)
                Exit For
            End If
        Next
    End If
Next
End If
Next

```

Fig.6-21: Removing items from a subscription



Removing items from a group should always be carried out for an entire list of items, so that the administrative tasks for login and logon of the group only have to be executed once in the OPC server.

Reading Control Data

General

This section describes read access to control data. A distinction is made between

- "synchronous" and
- "asynchronous" access.

The control data is read in the "Button_Read_Click" event handler of the "Read" button.

The relevant code segments are:

- "Step: 11" - synchronous reading
- "Step: 12" - asynchronous reading
- "Step: 13" - evaluate data

Synchronous reading

In case of a "synchronous" access, the client waits until the server has provided the data. Depending on the behavior defined in "Opc.Da.Item", the data is either read from the internal "cache" of the OPC server or from the control "device". The client does not accept **any further operating actions** during this period, i.e. the "synchronous" call is blocking.



Data is generally read more quickly from the "cache" than from the "device". This behavior should always be taken into consideration when designing an OPC client.

"Synchronous" reading ("**Step: 11**") of subscription data is effected via the "opcValueResults = Me.m_OpcServer.Subscriptions(0).Read()" function request and the transfer of the items to be read "Me.m_OpcServer.Subscriptions(0).Items" as parameters.

In this example the entire group is read and the result is forwarded to the "OnReadComplete" function for processing.

The "OnReadComplete" function is described in [chapter "Evaluating data" on page 192](#).

Asynchronous reading

With "asynchronous" access the task is transferred to the server and the function call returns immediately. The server processes the order and reports the result of the processed order to the client via a pre-defined "delegate" ("OnReadComplete" event handler). This means that the client can **accept and process further commands** via the server during order processing.

"Asynchronous" reading ("**Step: 12**") is effected, as for "synchronous" reading, via the "opcValueResults = Me.m_OpcServer.Subscriptions(0).Read()" function call.

The call differs only in the number of parameters to be transferred. The transfer parameters and their description are listed in the following table.

Transfer parameters	Description
Opc.Da.Item()	Item collection of all items to be read. As with synchronous reading.
Object	ID transferred from the client, which is given to the "delegate" when processing the results (optionally "Nothing").
ReadCompleteEventHandler	"Delegate" for processing the results of the enquiry. The "delegate" has to be transferred to the function request with the instruction "New ReadCompleteEventHandler(AddressOf OnReadComplete)".
IRequest	Handle to interrupt the request via the "Subscription.Cancel()" function call (optionally 'Nothing').

Fig. 6-22: Transfer parameter for asynchronous reading

Client Development

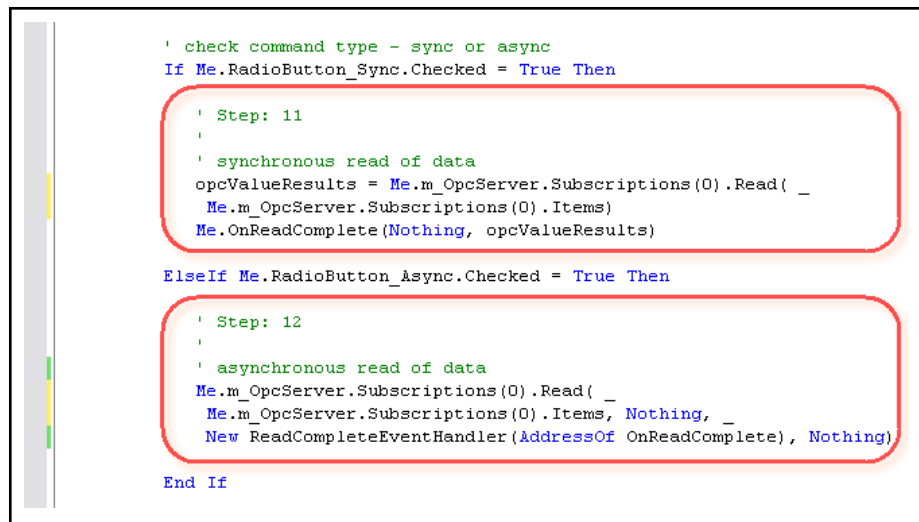


Fig.6-23: Synchronous and asynchronous reading of control data

Evaluating data

The "OnReadComplete" function for evaluating data returned from the OPC server is called following the "synchronous" reading from the OPC client and as an "event handler" ("Delegate") by the OPC server when providing the data following the "asynchronous" request.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread, it must be verified whether the "OnReadComplete" function call is running in the same thread as the thread of the form. This is verified in "Step: 13.1".

If the function is not called in the thread of the form, the call is synchronized by the "BeginInvoke()" function.

In "Step: 13.2" the results are evaluated and displayed in the list.

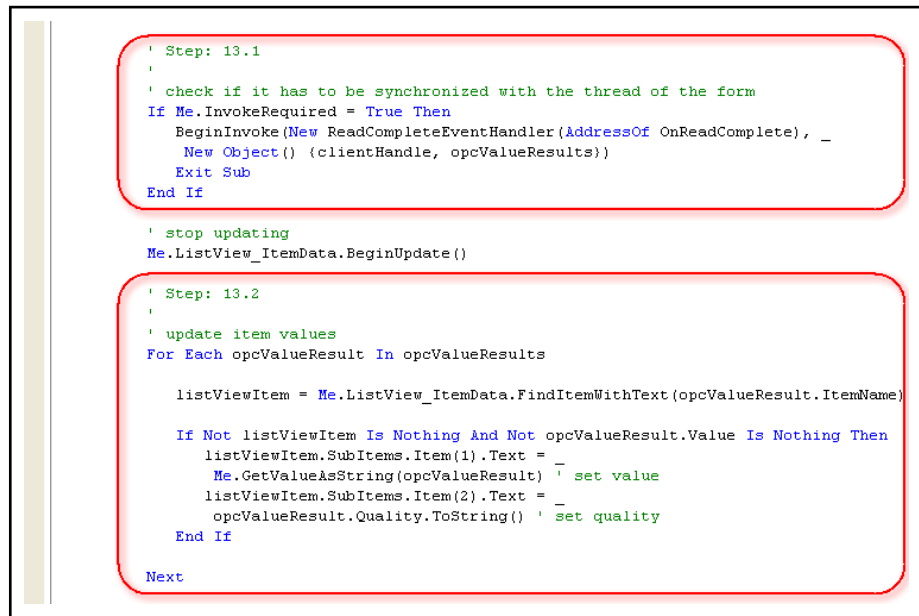


Fig.6-24: Evaluating data - reading

Writing control data

General

This section describes write access to control data. A distinction is made between

- "synchronous" and
- "asynchronous" access.

Control data is written in the "Button_Write_Click" event handler of the **"Write"** button.

The relevant code segments in the OPC client example project are:

- **"Step: 14"** - synchronous writing
- **"Step: 15"** - asynchronous writing
- **"Step: 16"** - evaluate data

Synchronous writing

With "synchronous" access the client waits until the server has completed the write task.

The data is written in the "cache" and in the "device" of the OPC server if the subscription and the item are active. The client does not accept **any further operating actions** during this period, i.e. the "synchronous" call is blocking.

"Synchronous" writing (**"Step: 14"**) of subscription data is effected via the "opcResults = Me.m_OpcServer.Subscriptions(0).Write()" function call and the transfer of the "Opc.Da.ItemValue" ("opcValues") to be written as parameters.

In this example only one item from the entire subscription is written and the write task result is forwarded to the "OnWriteComplete" function for processing.

The "OnWriteComplete" function is described in [chapter "Evaluating data" on page 194](#).

Asynchronous writing

With "asynchronous" access the task is transferred to the server and the function call returns immediately.

The server subsequently processes the task and reports the task processing result back to the client using a pre-defined "delegate" ("OnWriteComplete" event handler). That means the client can **accept and process further operating commands** via the server during task processing.

The "asynchronous" writing (**"Step: 15"**) is carried out via the function call as with "synchronous" reading

"opcResults = Me.m_OpcServer.Subscriptions(0).Write()".

The call differs only in the number of parameters to be transferred. The transfer parameters and their description are listed in the following table.

Transfer parameters	Description
Opc.Da.ItemValue()	Item collection of all items to be written with values. As with synchronous writing.
Object	ID transferred from the client, which is given to the "delegate" when processing the results (optionally "Nothing").

Client Development

Transfer parameters	Description
WriteCompleteEventHandler	"Delegate" for processing the results of the task. The "delegate" has to be transferred to the function request with the instruction "New WriteCompleteEventHandler(AddressOf OnWriteComplete)".
IRequest	Handle to interrupt the order via the "Subscription.Cancel()" function call (optionally "Nothing").

Fig. 6-25: Transfer parameters for asynchronous writing

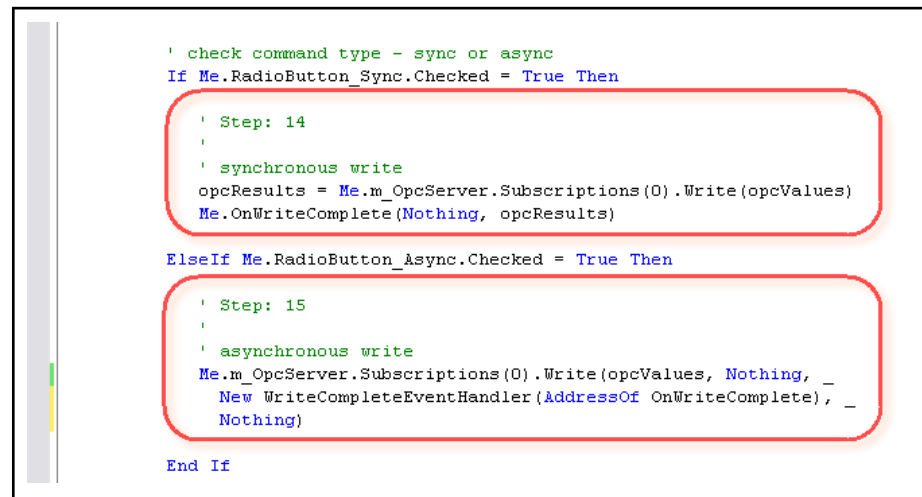


Fig. 6-26: Synchronous and asynchronous writing of control data

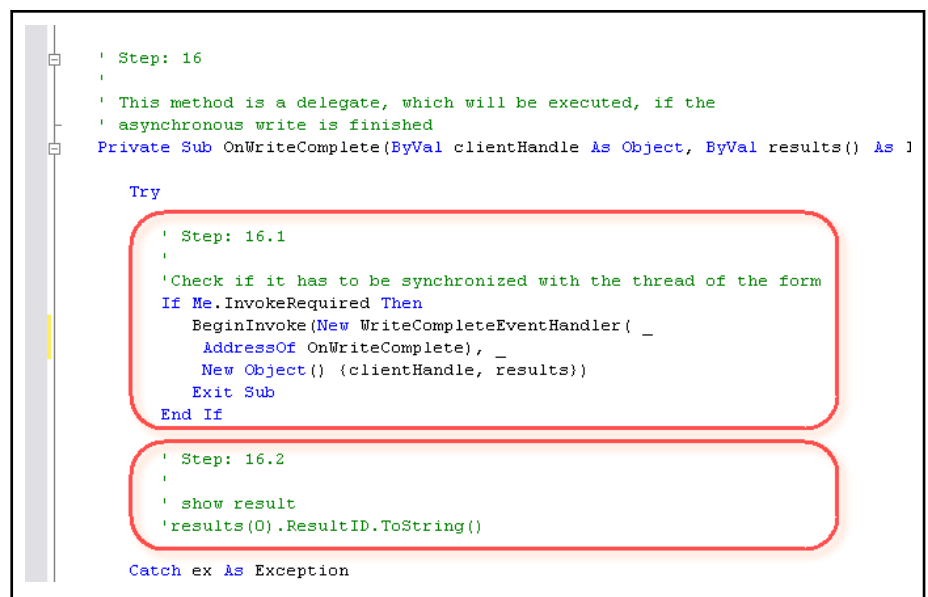
Evaluating data

The "OnWriteComplete" function serves to evaluate the results delivered back from the OPC server.

It is called by the OPC client after "synchronous" writing and as an "event handler" ("delegate") from the OPC server, with the completion of task processing following the "asynchronous" command.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread, it must be verified whether the "OnWriteComplete" function thread is running in the same thread as the thread of the form. This is verified in **"Step: 16.1"**. If the function is not called in the thread of the form the request is synchronized by the "BeginInvoke()" function.

In **"Step: 16.2"** the results of the write task can be evaluated. This evaluation can be performed in a "log file" or e.g. in a "StatusStrip" control element of the application. The example client does not contain an evaluation of the results of the write task.



```

' Step: 16
'
' This method is a delegate, which will be executed, if the
' asynchronous write is finished
Private Sub OnWriteComplete(ByVal clientHandle As Object, ByVal results() As I

    Try

        ' Step: 16.1
        '
        ' Check if it has to be synchronized with the thread of the form
        If Me.InvokeRequired Then
            BeginInvoke(New WriteCompleteEventHandler( _
                AddressOf OnWriteComplete), _
                New Object() {clientHandle, results})
            Exit Sub
        End If

        ' Step: 16.2
        '
        ' show result
        ' results(0).ResultID.ToString()

    Catch ex As Exception

```

Fig. 6-27: Evaluating data - writing

Cyclic Reading of Control Data

General

This section describes cyclic reading of control data.

The prerequisite for cyclic reading of values is that

- the subscription and
- the item are active and that
- an event handler is logged on to the subscription.

Activation and deactivation of cyclic reading is controlled via the "RadioButton_CheckedChanged" event of the "" "Synchronous", "Asynchronous" and "Cyclic" radio buttons.

The relevant code segments in the OPC client example project are:

- "Step: 17" - activate cyclic reading
- "Step: 18" - deactivate cyclic reading
- "Step: 19" - evaluate data

Activating cyclic reading

As the prerequisites of the active subscription and active items have already been met, the "DataChanged" event handler only has to be added to the "iSubscription.DataChanged" subscription list ("Step: 17"). Adding the event handler fulfills all prerequisites and the client receives the "DataChanged" events.

Deactivating cyclic reading

To deactivate cyclic reading the event handler

"DataChanged" is removed from the subscription list

"iSubscription.DataChanged" again. Once it has been removed, no further value changes can be reported to the client.

Client Development

```

If radioBtn.Text = Me.RadioButton_Cyclic.Text Then

    If Me.RadioButton_Cyclic.Checked = True Then

        ' Step: 17
        '
        ' add event handler for cyclic communication
        iSubscription = Me.m_OpcServer.Subscriptions(0)
        AddHandler iSubscription.DataChanged, AddressOf DataChanged

    Else

        ' Step: 18
        '
        ' remove event handler for cyclic communication
        iSubscription = Me.m_OpcServer.Subscriptions(0)
        RemoveHandler iSubscription.DataChanged, AddressOf DataChanged

    End If

End If

```

Fig.6-28: Adding and removing event handling

Evaluating data

The "DataChanged" function is requested cyclically by the OPC server at the time intervals specified in the "UpdateRate" element of the "Opc.Da.SubscriptionState" class.

All items are transferred to the "DataChanged" function with the corresponding values if

- the value has changed or
- the maximum "age" of the value was exceeded.

This means it is dependent on the configuration of the "MaxAge" and "MaxAgeSpecified" elements of the "Opc.Da.Item" class.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread, it has to be verified whether the "DataChanged" function thread is running in the same thread as the thread of the form. This is verified in "Step: 19.1".

If the function is not called in the thread of the form, the call is synchronized by the "BeginInvoke()" function.

In "Step: 19.2" the results are evaluated and displayed in the list view as with synchronous and asynchronous reading.

```

Try

    ' Step: 19.1
    '
    ' check if it has to be synchronized with the form thread
    If (Me.InvokeRequired = True) Then
        Me.BeginInvoke(New DataChangedEventHandler(AddressOf DataChanged), _
            New Object() {subscriptionHandle, requestHandle, opcValueResults})
        Exit Sub
    End If

    ' stop updating
    Me.ListView_ItemData.BeginUpdate()

    ' Step: 19.2
    '
    ' update item values
    Me.OnReadComplete(Nothing, opcValueResults)

End Try

```

Fig.6-29: Evaluating data - cyclic reading

Error Handling

Only a minimum of error handling was implemented with the example OPC client. Errors which arise with an OPC client used on production machines should be evaluated and recorded in more detail.

The OPC client is therefore restricted to verifying variables and simple exception handling in the form of

- "If" queries and
- exception handling with "Try...Catch..." instructions

"If" queries

"If" queries verify the parameters required to execute the function. If the specified definitions cannot be fulfilled the function will be interrupted at this point.

Exception handling with "Try...Catch..." instructions

As the majority of components report an error to the caller during execution of a function by "throwing" an exception, at least one "Try...Catch..." statement should be included in every function/method. In the example OPC client only one "Try...Catch..." instruction was programmed due to the scope of the individual functions.

6.2 OPC Client Visual C# .NET

6.2.1 General

This section deals with the development of the Visual C# .NET OPC Client using "Microsoft Visual Studio .NET 2005" and covers the following topics:

This section deals with the following topics:

- Creating projects
- Adding References
- Adding Windows forms and control elements
- OPC client implementation.

The following diagram shows the OPC Client as an active Windows application.

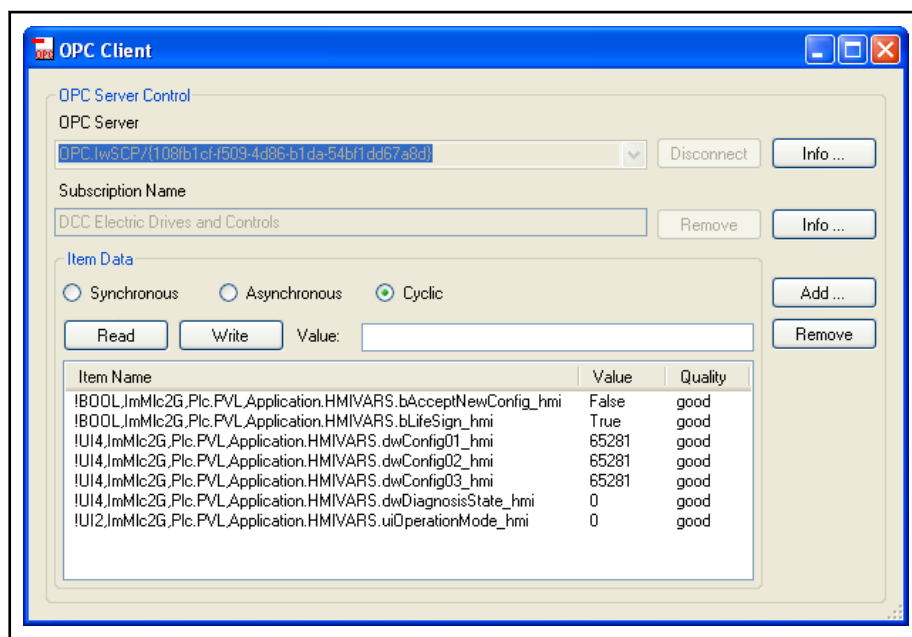


Fig. 6-30: OPC client interface

Client Development

6.2.2 Visual C# Project "OPC Client "

New Project

To create a new project in the "Microsoft Visual Studio .NET 2005" development environment, either click on the link "New Project" on the start page or in the menu **File ► New** click on the menu item **Project....**

The "New Project" dialog will appear, as shown below.

Project type and template can be selected here:

- "Other languages \ Visual C# \ Windows" or
- "Visual C# \ Windows" and
- "Windows application"

Enter "OPCClient" as the project name and adapt the suggested location accordingly.

The project is created by clicking on **OK**.

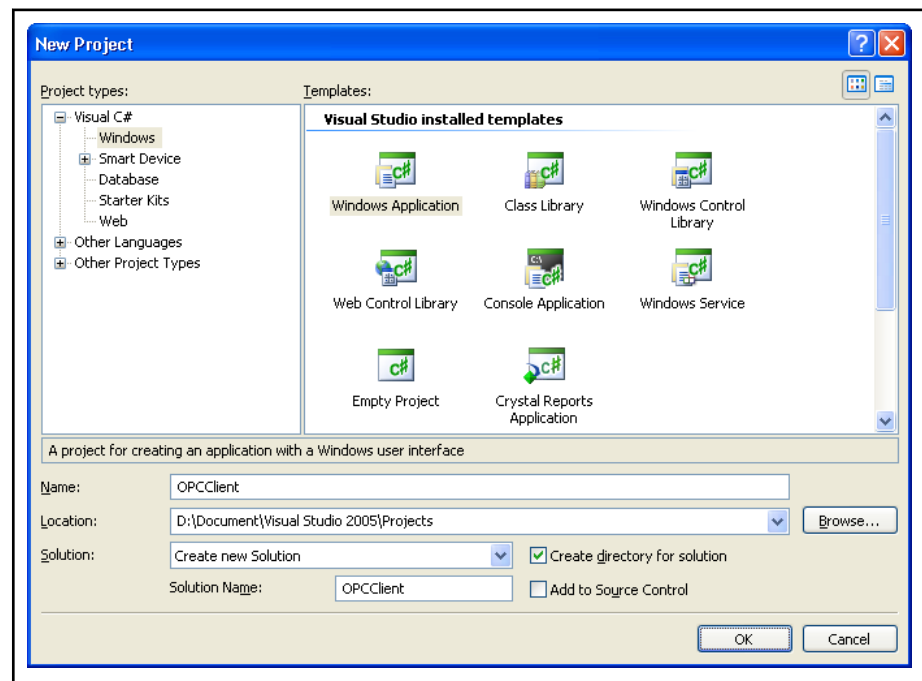


Fig.6-31: Create New Project

Adding References

The project has to contain a reference to "OpcNetApi.dll" and "OpcNetApi.Com.dll" to allow access to the OPC .NET API functionalities.

The "project references" are managed in the project folder explorer on the node "references" of the project "OPCClient" or using the references already added.

References are added with the menu item **Add References ...** from the respective context menu or with the menu **Project** . In the "Add Reference" dialog, the two files "OpcNetApi.dll" and "OpcNetApi.Com.dll" can be added to the project on the "Browse" tab.

Both files are provided with the "IndraWorks Suite" installation and are saved in the installation folder (e.g. "C:\Program Files\Rexroth\IndraWorks").

By opening the files they are applied to the node list "References:" .

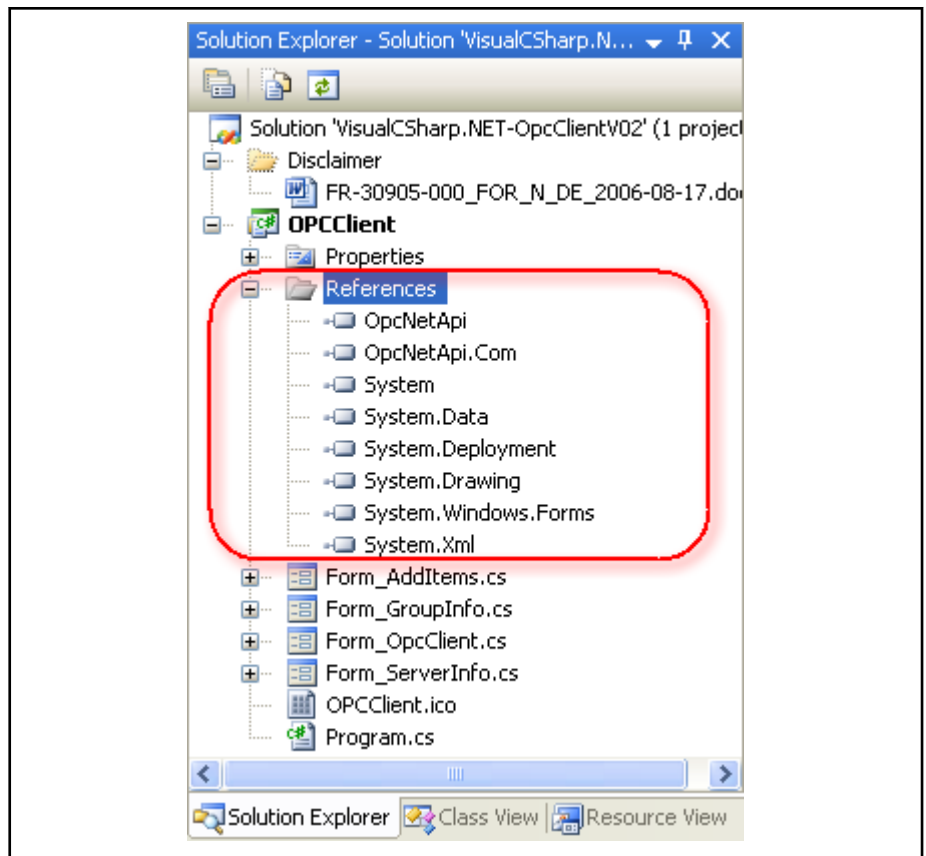


Fig. 6-32: Project references

Adding Windows Form and Control Elements

Add a windows form to the start project using the menu item **Project ► Add Windows Form....**

In the "Add New Element - OPCClient" dialog the "Windows form" template has already been selected and only the name of the form needs to be entered.

All Windows forms created in the OPC client example have the name of the identifier "Form_" as a prefix.

The following figure shows an overview of the "OPC Client" project in the "Solution Explorer".

Client Development

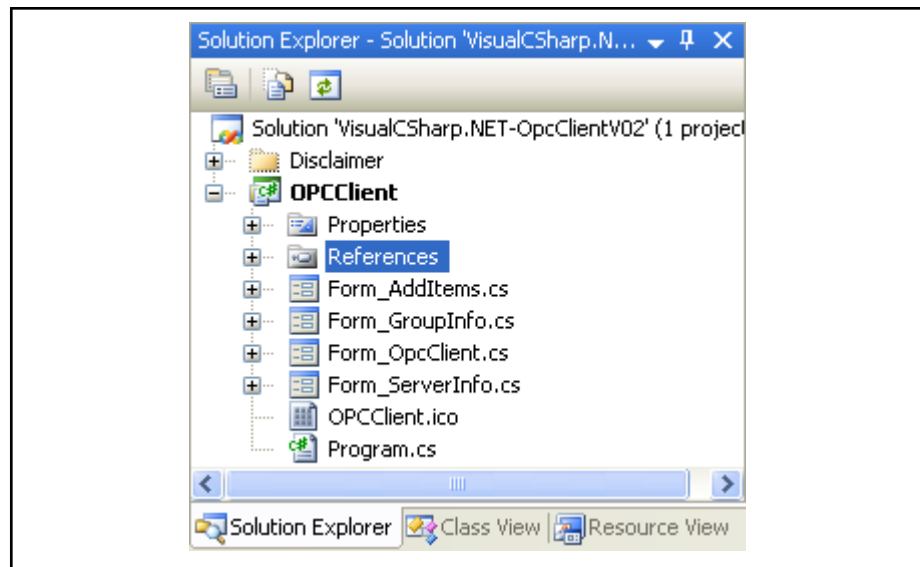


Fig.6-33: Project overview

"Windows forms" are designed with the "Windows Forms Designer" and the control elements available in the "Toolbox". Select the control elements and place them on the form.

The control elements for the selected object are configured in the "Properties" window.

Size and position of the control elements can also be changed using the mouse in the Windows Forms Designer.

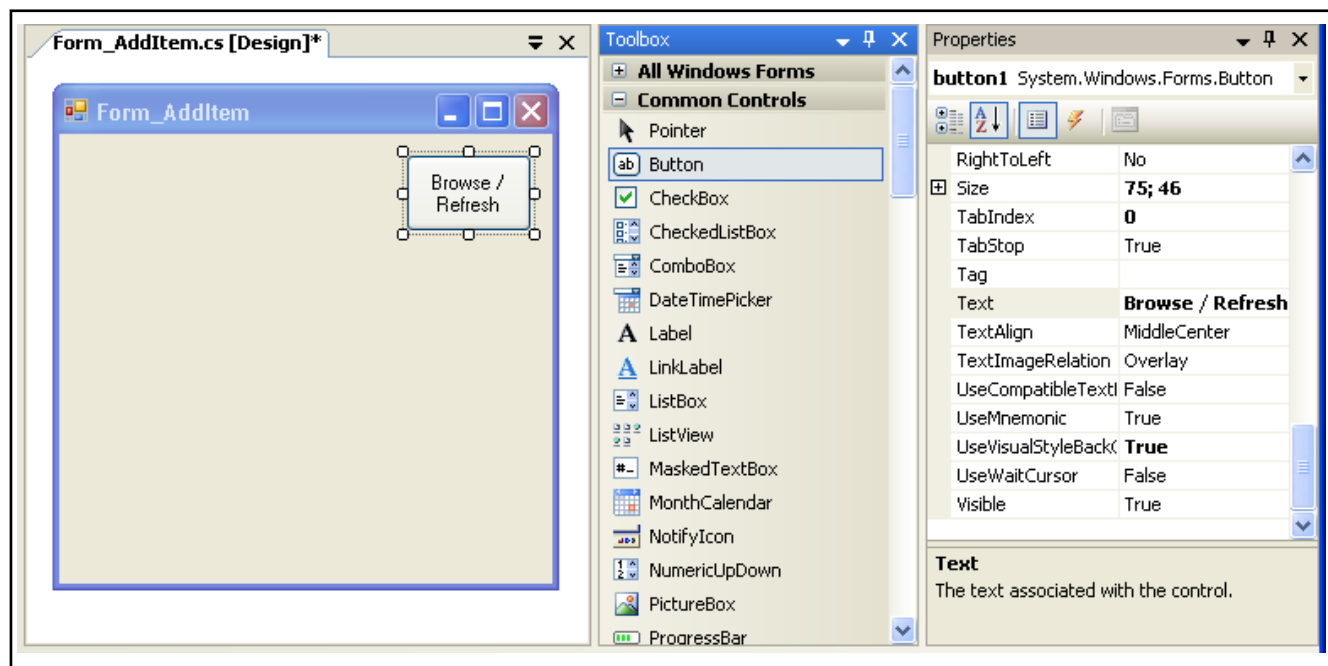


Fig.6-34: Adding and configuring control elements

The following figure shows the "OPC Client" form in "Windows Forms Designer".

To maintain clarity we will not provide a detailed description of the form and the individual control elements. All configurable properties can be taken from the example project.

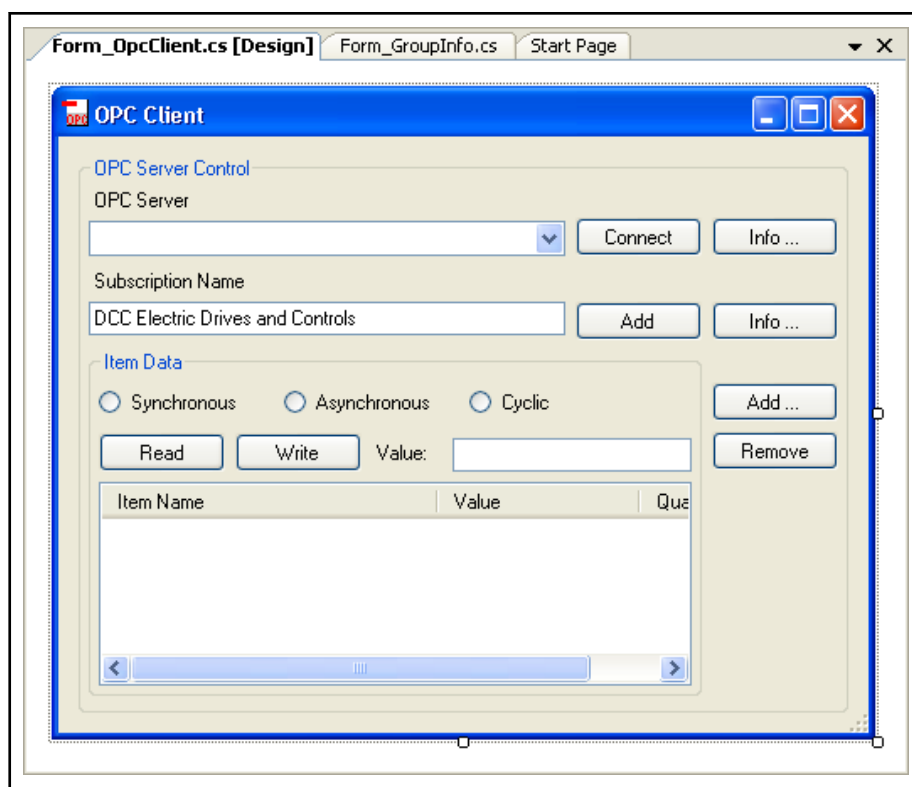


Fig. 6-35: "OPC Clients" Windows Forms Designer

Implementing OPC Client

General

The following section describes the procedure for programming the OPC client.

To provide better clarity in the program, individual regions of the class have been combined with the "#region" directive. These areas can be expanded or collapsed when creating the program.

The figure below shows the defined regions using the example of the "Form_OpcClient" form.

Range	Description
"#region" -"#endregion"	
"Declaration"	Contains declaration of the class (member variables and constants).
"Form Events"	Procedures for handling the events triggered by the form and the control elements contained in it.
"OPC Events"	Procedures for handling the events triggered by the OPC server.
"Additional Internal Functions"	Additional internal definitions, structures and functions for optimizing user guidance.

Fig. 6-36: Program structuring using "#region" directives

The relevant program lines in the example project are also identified by the keyword **"Step:"** and numbered consecutively. This means that using the search "Search in Files" (<Ctrl>+<Shift>+<F>) in Visual Studio allows direct

Client Development

branching to the implementation described by entering the search word e.g. **"Step: 2"**.

The implementation described in the following has been documented following the example of the operating order of the "OPC client" example application.

The structure is shown in the table below.

Chapter title	Description
chapter "Declaring Variables" on page 202	Declaration of the variables for managing OPC server and the Windows form.
chapter "Listing OPC Server" on page 203	Listing of all OPC servers installed on the local PC.
chapter "Creating and Deleting OPC Server Instance" on page 203	Creating and canceling the OPC server instance and displaying the OPC server information.
chapter "Creating and Canceling OPC Server Subscription" on page 206	Creating and canceling an OPC subscription and displaying the subscription information.
chapter "Listing Control Items" on page 208	Reading and displaying all control items (browsing)
chapter "Adding and Removing OPC Items" on page 209	Adding and deleting OPC items.
chapter "Reading Control Data" on page 212	Reading "synchronous" and "asynchronous" control data.
chapter "Writing Control Data" on page 214	Writing "synchronous" and "asynchronous" control data.
chapter "Cyclic Reading of Control Data" on page 216	Cyclic Reading of Control Data

Fig.6-37: Overview of implementation structure

Declaring Variables

In the **"Step: 1"** declaration section the "Form_OpcClient" variables are declared.

The chosen syntax for the names provides writing of all class variables with the prefix "m_" ("member variables") and "capitalizing" the constant definitions.

The variables are

- the object for the OPC server instance and
- variables for managing the form.

The line **"private Opc.Da.Server m_OpcServer = null;"** is the declaration relevant to the example. The "m_OpcServer" variable contains the created OPC server object.

```

// Step: 1
//
// declaration of members used for opc administration
#region " Declaration "

// Step: 1
//
// declaration of members used for opc server administration
private Opc.Da.Server m_OpcServer = null;

//
// members to store configuration
private OPCClient.Form_AddItems m_DialogAddItems = null;
private InternalStateControl m_InternalStateControl;

//
// constant values
private readonly String BUTTON_DESCRIPTION_SERVER_CONNECT = "Connect";
private readonly String BUTTON_DESCRIPTION_SERVER_DISCONNECT = "Disconnect";
private readonly String BUTTON_DESCRIPTION_GROUP_ADD = "Add";
private readonly String BUTTON_DESCRIPTION_GROUP_REMOVE = "Remove";
private readonly Int32 MAX_ITEMS = 10;

#endregion

```

Fig. 6-38: Declaration of class variables

Listing OPC Server

All OPC servers installed locally on the PC are listed in the "Load Event" "Step: 2" of the "Form_OpcClient" form.

All OPC servers are requested via the "opcDiscovery.GetAvailableServers()" method of the Discovery interface "Opc.IDiscovery" of the "OpcEnum.exe" service, which support the specification "Data Access 2.0". The available OPC specifications are defined in the structure "Opc.Specification".

The list provided by the Discovery interface contains the complete URL path with "ProgID" (without version index) and "GUID" to clearly identify the server.

```

// Step: 2
//
// browse for available opc servers on this system
private void Form_OpcClient_Load ( object sender, EventArgs e )
{
    try
    {
        // Step: 2
        //
        // browse for available opc servers on this system
        Opc.IDiscovery opcDiscovery = new OpcCom.ServerEnumerator ();
        Opc.Server[] opcServers = opcDiscovery.GetAvailableServers (
            Opc.Specification.COM_DA_20);

        // copy opc server to ...
        this.ComboBox_Server.BeginUpdate ();

        foreach ( Opc.Server opcServer in opcServers )
        {
            this.ComboBox_Server.Items.Add ( opcServer.Url.Path );
        }

        catch ( Exception except )
        {
        }
    }
}

```

Fig. 6-39: Listing installed OPC servers

Creating and Deleting OPC Server Instance

General

This section describes how to create and cancel the OPC server instance and how to read the server information. The OPC Server instance is man-

Client Development

aged in the event handler "Button_Connect_Disconnect_Click" of the button "**Connect**" or "**Disconnect**", reading and displaying the server information is carried out in the function "Button_ServerInfor_Click" of the "**Info ...**" button.

The relevant code segments in the OPC client example project are:

- "**Step: 3**" - create OPC server instance
- "**Step: 4**" - delete OPC server instance
- "**Step: 5**" and "**Step: 21**" - read server information

Creating OPC server instance - Connect

The creation of an OPC server instance and the connection establishment is described in "**Step: 3**".

The first step creates an OPC server instance "New Opc.Da.Server" and transfer of the parameters "OpcCom.Factory" and the "Opc.URL".

The "Opc.URL" contains information about the OPC specification "Opc.UrlScheme.DA" used, the PC name "localhost" or the IP address on which the server is to be started, as well as the server identifier. In the example project, the server identifier is read from the combination list field "ComboBox_Server".

The following options are possible for specifying the server identifier:

- "**OPC.lwSCP**"
ProgID without version index
- "**OPC.lwSCP.1**"
ProgID with version index
- "**OPC.lwSCP/{108fb1cf-f509-4d86-b1da-54bf1dd67a8d}**"
ProgID without version index and GUID
- "**OPC.lwSCP.1/{108fb1cf-f509-4d86-b1da-54bf1dd67a8d}**"
ProgID with version index and GUID

The OPC server is instanced by calling the function "this.m_OpcServer.Connect()".

Deleting OPC server instance - Disconnect

Deletion of the OPC server instance is shown in "**Step: 4**".

Calling the function "this.m_OpcServer.Disconnect()" causes log-off from the OPC server. If no further client is connected to the server at this point, it will be terminated.

Subsequently the function "this.m_OpcServer.Dispose()" will be called to release resources which are held in the OPC server instance and the instance is canceled by "this.m_OpcServer = null".

```
try
{
    // show hour glas
    this.Cursor = Cursors.WaitCursor;

    if ( this.Button_Connect_Disconnect.Text.Equals (BUTTON_DESCRIPTION)
    {
        // Step: 3
        //
        // select opc server and
        this.m_OpcServer = new Opc.Da.Server ( new OpcCom.Factory(),
        new Opc.URL(Opc.UrlScheme.DA + "://localhost/" +
        this.ComboBox_Server.Text) );

        // connect with ...
        this.m_OpcServer.Connect();

        // Enable or disable controls
        this.ComboBox_Server.Focus();
        this.SetInternalStateControl(InternalStateControl.Connected);
    }
    else if ( this.Button_Connect_Disconnect.Text.Equals (BUTTON_DESCRIP
    {
        // Step: 4
        //
        // disconnect from server and clean up
        this.m_OpcServer.Disconnect();
        this.m_OpcServer.Dispose();
        this.m_OpcServer = null;

        // Enable or disable controls
        this.SetInternalStateControl(InternalStateControl.Disconnected);

        // clear browse info
        if ( this.m_DialogAddItems is Form_AddItems )
            this.m_DialogAddItems.Clear();
    }
    }
}
```

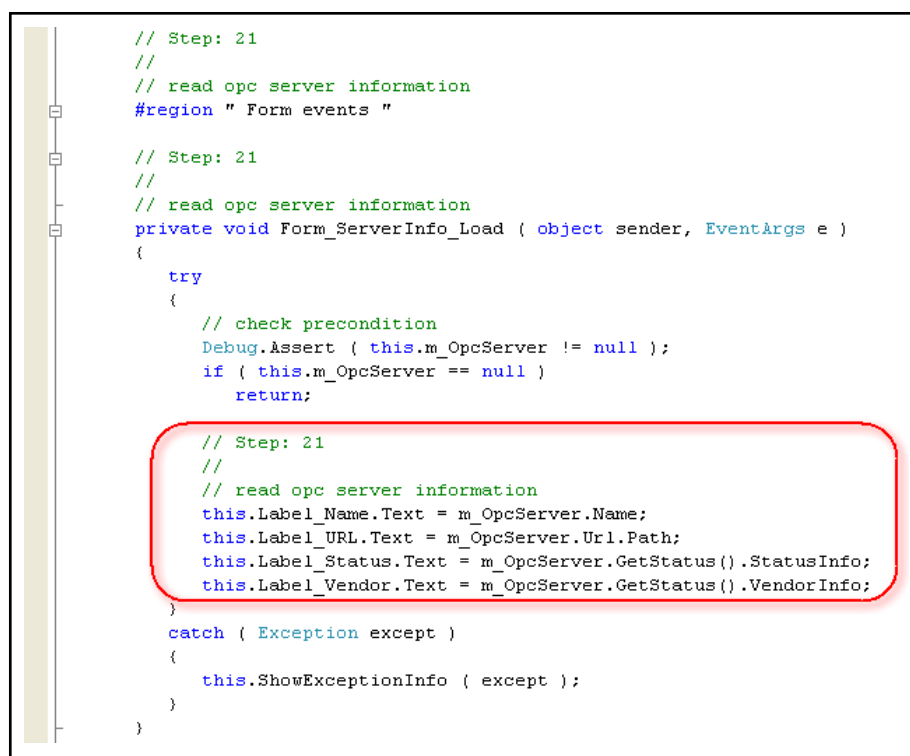
Fig. 6-40: Creating and canceling OPC server instance

Reading server information

The information provided by the OPC Server is read and displayed in the "Load Event" **"Step: 21"** of the form "Form_ServerInfo".

The form is called in **"Step: 5"** (no figure).

Client Development



```

// Step: 21
//
// read opc server information
#region " Form events "

// Step: 21
//
// read opc server information
private void Form_ServerInfo_Load ( object sender, EventArgs e )
{
    try
    {
        // check precondition
        Debug.Assert ( this.m_OpcServer != null );
        if ( this.m_OpcServer == null )
            return;

        // Step: 21
        //
        // read opc server information
        this.Label_Name.Text = m_OpcServer.Name;
        this.Label_URL.Text = m_OpcServer.Url.Path;
        this.Label_Status.Text = m_OpcServer.GetStatus().StatusInfo;
        this.Label_Vendor.Text = m_OpcServer.GetStatus().VendorInfo;
    }
    catch ( Exception except )
    {
        this.ShowExceptionInfo ( except );
    }
}

```

Fig. 6-41: Reading OPC server information

Creating and Canceling OPC Server Subscription

General

This section describes login and logout of a "subscription" (group) on the OPC server. The subscription is managed in the "Button_Add_Remove_Group_Click" event handler of the "Add" or "Remove" button and the subscription information is read and displayed in the "Button_GroupInfo_Click" function of the "Info ..." button.

The relevant code segments in the OPC client example project are:

- "Step: 6" - create subscription
- "Step: 7" - cancel subscription
- "Step: 8" and "Step: 22" - display subscription information

Creating a subscription - CreateSubscription

A subscription is created in the OPC server collection with the function "this.m_OpcServer.CreateSubscription()" and transfer of the parameter "Opc.Da.SubscriptionState" ("Step: 6", see [fig. 6-43 "Creating an OPC subscription" on page 207](#)).

The elements of the "Opc.Da.SubscriptionState" class serve to clearly identify the subscription and to configure it in relation to its behavior or processing by the OPC server.

The following table provides a description of the class elements.

Class element	Description
Active	The control cyclically reads the subscription items and saves them in the server cache.
ClientHandle	"Unique Identifier" (issued by the client).

Class element	Description
Deadband	Smallest possible percentage change of a value to trigger an event to the client.
KeepAlive	Largest possible time interval between two events to the client (life sign).
Locale	Definition of localization in the format "[language Code]-[country/region code]", e.g. "de-DE", "en-US"
Name	"Unique Name" (issued by the client). Specified by the "TextBox_Group.Text" text box.
ServerHandle	"Unique Identifier" (issued by the server).
UpdateRate	Time interval during which the server checks the data and sends an event to the client.

Fig. 6-42: Elements of class "Opc.Da.SubscriptionState"

```

// Step: 6 / 7
//
// create and cancel subscription
private void Button_Add_Remove_Group_Click ( object sender, EventArgs e )
{
    try
    {
        // show hour glas
        this.Cursor = Cursors.WaitCursor;

        if ( this.Button_Add_Remove_Group.Text.Equals ( BUTTON_DESCRIPTION_GROUP_ADD ) )
        {
            // Step: 6
            //
            // create subscription
            Opc.Da.SubscriptionState subSubscriptionState = new Opc.Da.SubscriptionState();
            subSubscriptionState.Active = true; // activate cyclic communication
            subSubscriptionState.ClientHandle = System.Guid.NewGuid().ToString();
            subSubscriptionState.Deadband = 0.0F;
            subSubscriptionState.Name = TextBox_Group.Text;
            subSubscriptionState.UpdateRate = 1000;

            this.m_OpcServer.CreateSubscription(subSubscriptionState);

            // Enable or disable controls
            this.SetInternalStateControl ( InternalStateControl.SubscriptionSet );
        }
        else if ( this.Button_Add_Remove_Group.Text.Equals ( BUTTON_DESCRIPTION_GROUP_REP
    {

```

Fig. 6-43: Creating an OPC subscription

Canceling a subscription - CancelSubscription

A subscription can be canceled by calling the function

"this.m_OpcServer.CancelSubscription()" with transfer of the parameter "Opc.Da.ISubscription" ("Step: 7").

Once the subscription has been canceled from the OPC server collection it is essential that the "iSubscription.Dispose()" function is called so that the resources used by the subscription can be released again. The reference of the interface variables is canceled by assigning "iSubscription = null".

The local allocation "iSubscription = this.m_OpcServer.Subscription()" performed at the start of the code segment is required, as the subscription was removed from the OPC server collection following calling of function "this.m_OpcServer.CancelSubscription()" and is no longer available.

Client Development

```

    }
    else if ( this.Button_Add_Remove_Group.Text.Equals ( BUTTON_DESCRIPTION_GROU
    {
        // Step: 7
        //
        // cancel subscription and clean up
        Opc.Da.ISubscription iSubscription = this.m_OpcServer.Subscriptions[0];
        this.m_OpcServer.CancelSubscription(iSubscription);
        iSubscription.Dispose();
        iSubscription = null;

        // Enable or disable controls
        this.SetInternalStateControl ( InternalStateControl.Connected );
    }
}
catch ( Exception except )

```

Fig.6-44: Canceling OPC subscription

Displaying subscription information

The information provided by the subscription is read and displayed in the "Load Event" ("Step: 22") in the form "Form_GroupInfo".

The form is called in "Step: 8" (no figure).

```

// check precondition
Debug.Assert ( this.m_OpcServer != null );
if ( this.m_OpcServer == null )
    return;

// Step: 22
//
// read opc subscription information
Opc.Da.Subscription subSubscription = m_OpcServer.Subscriptions[0];
this.Label_Name.Text = subSubscription.Name;
this.Label_Deadband.Text = subSubscription.GetState ( ).Deadband.ToString ( ) + " %";
this.Label_UpdateRate.Text = subSubscription.GetState ( ).UpdateRate.ToString ( ) + " ms";
}
catch ( Exception except )

```

Fig.6-45: Reading OPC subscription information

Listing Control Items

General

This section describes the reading of all items configured in the control for access via the OPC server.

The relevant code segments in the OPC client example project are:

- "Step: 20", "Step: 20.1" and "Step: 20.2" - reading control items

Reading control items

Clicking on the "Add ..." button on the "Form_OpcClient" form calls the "Form_AddItems" form. As it may possibly take some time to read the items, the items available on the OPC server are not read in the "Load Event" of the form but by using the "Button_BrowseItems_Click" function which is activated by clicking on the "Browse / Refresh" button.

The two classes "Opc.ItemIdentifier" and "Opc.Da.BrowseFilters" of the "this.m_OPCServer.Browse()" function are transferred as parameters to read the items ("Step: 20.2").

The classes have the following function:

Opc.ItemIdentifier	The "Opc.ItemIdentifier" class specifies the "Starting point" of the search in the "tree hierarchy" of the items. This means that if, for example, a search is interrupted, the search can be continued again at this point.
Opc.Da.BrowseFilter	The class "Opc.Da.BrowseFilter" serves as a filter criterion for the result of the function call. There is an option here to only read variables, which begin e.g. with a certain sequence of characters.

As there is no default specification for the format in which an OPC server returns the "browsing" information, reading of the control items was divided into two functions ("Step: 20.1" and "Step: 20.2"). This division allows reading of flat hierarchies (tree structure without sub-nodes) and structured hierarchies (tree structure with sub-nodes) (see following figures).

```
// clear list, stop updating
this.ListBox_Available_Items.BeginUpdate ( );
this.ListBox_Available_Items.Items.Clear ( );

// Step: 20.1
//
// browse items
Opc.Da.BrowseFilters filters = new Opc.Da.BrowseFilters ( );
filters.ElementNameFilter = "";
filters.BrowseFilter = Opc.Da.browseFilter.all;
filters.ReturnAllProperties = false;
filters.ReturnPropertyValues = false;

Opc.ItemIdentifier itemID = new Opc.ItemIdentifier ( );
itemID.ItemName = "";

this.BrowseItems ( itemID, filters );
}
catch ( Exception except )
```

Fig.6-46: Reading control items - filter configuration

```
// Step: 20.2
//
// browse items
opcBrowseElements = this.m_OpcServer.Browse ( itemID, filters, out opcBrowsePosition );

// adds each element to the list
foreach (Opc.Da.BrowseElement opcBrowseElement in opcBrowseElements)
{
    //
    // avoid errors while using .NET API 1.30
    if ( opcBrowseElement.ItemName != null )
        itemName = opcBrowseElement.ItemName;
    else
        itemName = opcBrowseElement.Name;

    this.ListBox_Available_Items.Items.Add ( itemName );

    // check on subordinate items
    if (opcBrowseElement.HasChildren == true)
    {
        // set branch to search
        opcItemID.ItemName = itemName; // opcBrowseElement.ItemName;
        this.BrowseItems ( opcItemID, filters );
    }
}
```

Fig.6-47: Reading control items - recursion

Adding and Removing OPC Items

General

This section describes the adding and deleting of items to/from a subscription.

The relevant code segments in the OPC client example project are:

- "Step: 9", "Step: 9.1" and "Step: 9.2" - adding items
- "Step: 10" - deleting items

Adding items - AddItems

In the form event "Button_Add_Item_Click" ("Step: 9") the form "Form_AddItems" is called and the items selected in the form are added.

Client Development

The variables can be entered separately in the text field of the "Add Items" form by ";" or they can be selected from the list. When entering the variables, the item syntax of the respective OPC server is to be taken into account.

The following shows the item syntax for the "OPC.lwSCP" server and the "IndraLogic OPC" server.

"OPC.lwSCP" server

- **PLC variables**

"![required data type],[control name or alias],[lwSCP service],[fully qualifying variable name]"

Example:

"!Bool,ImMlc2G,Plc.PVL,Application.HMIVARS.bLifeSign_hmi"

- **sercos parameter**

"![required data type],[control name or alias],[lwSCP service],[drive address], [parameter number], [element coding]"

Example:

"!UI4,ImMlc2G,SPA1,0,C-0-0001,D"

"!UI4,ImMlc2G,SPA1,0,C-0-0001,40"



The "OPC.lwSCP" server allows access to parameters and PLC variables!



Further information on the item syntax of the "OPC.lwSCP" server can be found in [chapter "lwSCP and OPC.lwSCP Items" on page 89](#).

"IndraLogic OPC" server

- **PLC variables**

"[Control name].[fully qualifying variable name]"

Example:

"PLC_IL2G.Application.HMIVARS.bLifeSign_hmi"



The "IndraLogic OPC" server allows access to PLC variables.

The description of the "Browse" functionality is described in [chapter "Listing Control Items" on page 208](#).

Once the items have been entered or selected in the "Add Items" form, in **"Step: 9.1"** an "Opc.Da.Item" class is created and initialized for every item. The name of the item is assigned in the line "opcltems(index).ItemName = items(index)".

The class elements of class "Opc.Da.Item" used in the example are shown in the following table:

Class element	Description
Active	"DataChanged" events for the item are send to the client.
ActiveSpecified	The "Active" property was defined.
ItemName	Fully qualified item name. The syntax is dependent on the OPC server.
ItemPath	Fully qualified item path. This property is not evaluated by the "OPC.lwSCP".

Class element	Description
MaxAge	When the item is read from the OPC server cache the value may not be older than defined in "MaxAge".
MaxAgeSpecified	The "MaxAge" property was defined. If this property is set to "True" the value is read from the OPC server "cache" or from the "device" as a function of "MaxAge". If the property is set to "False" the value is always read from the "device".

Fig. 6-48: Elements of class "Opc.Da.Item"

Once all OPC items have been created the OPC item array is logged in to the subscription on the OPC server with the function request "this.m_OpcServer.Subscriptions[0].AddItems()" (**"Step: 9.2"**).

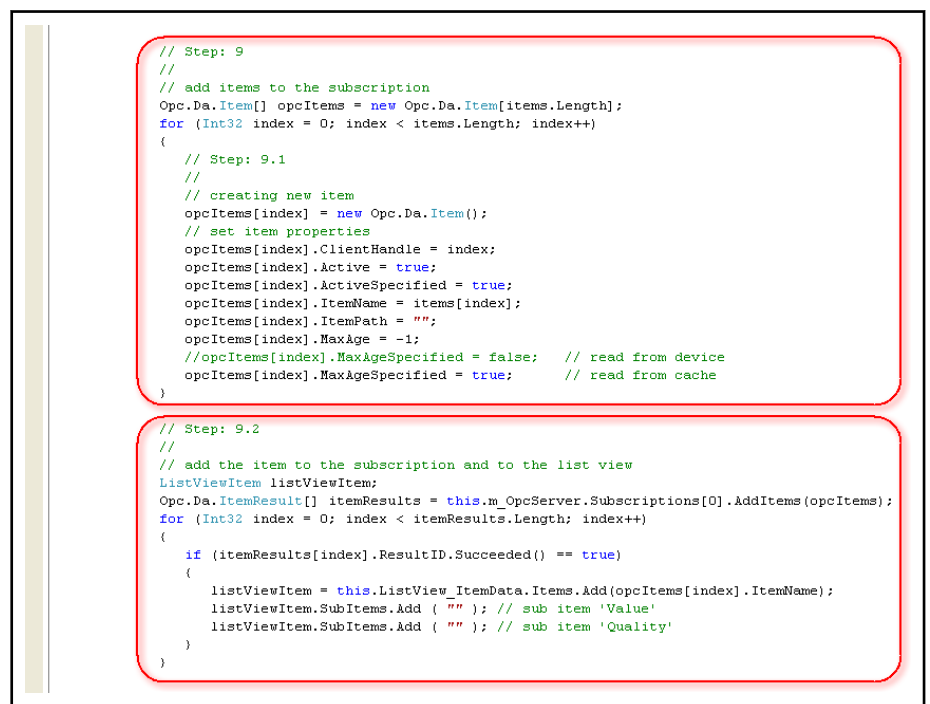


Fig. 6-49: Adding items to a subscription

Removing items - RemoveItems

To remove items from the OPC subscription, select the items from the list and click on **"Remove"**.

In the subsequently triggered "Button_Remove_Item_Click" form event the function "this.m_OpcServer.Subscriptions[0].RemoveItems()" is called, which removes the OPC item transferred as a parameter (**"Step: 10"**).

Client Development

```

// Step: 10
//
// remove selected items from opc subscription and from the list
for ( Int32 index = 0; index < selectedItems; index++ )
{
    foreach (Opc.Da.Item opcItem in this.m_OpcServer.Subscriptions[0].Items)
    {
        if ( this.ListView_ItemData.SelectedItems[index].Text.Equals ( opcItem.ItemName ) )
        {
            opcItemIds[index] = opcItem;
            break;
        }
    }
}
// remove items
Opc.IdentifiedResult[] opcIdentResult =
    this.m_OpcServer.Subscriptions[0].RemoveItems ( opcItemIds );

// clean up list view control
for ( Int32 index = 0; index < opcIdentResult.Length; index++ )
{
    if ( opcIdentResult[index].ResultID == ResultID.S_OK )
    {
        foreach ( ListViewItem listViewItem in this.ListView_ItemData.SelectedItems )
        {
            if ( opcIdentResult[index].ItemName.Equals ( listViewItem.Text ) )
            {
                this.ListView_ItemData.Items.Remove ( listViewItem );
                break;
            }
        }
    }
}

```

Fig. 6-50: Removing items from a subscription



Removing items from a group should always be carried out for an entire list of items, so that the administrative tasks for login and logon of the group only have to be executed once in the OPC server.

Reading Control Data

General

This section describes read access to control data.

A distinction is made between

- "synchronous" and
- "asynchronous" access.

The control data is read in the "Button_Read_Click" event handler of the "Read" button.

The relevant code segments are:

- **"Step: 11"** - synchronous reading
- **"Step: 12"** - asynchronous reading
- **"Step: 13"** - evaluate data

Synchronous reading

In case of a "synchronous" access, the client waits until the server has provided the data. Depending on the behavior defined in "Opc.Da.Item", the data is either read from the internal "cache" of the OPC server or from the control "device". The client does not accept **any further operating actions** during this period, i.e. the "synchronous" call is blocking.



The data is generally read more quickly from the "cache" than from the "device". This behavior should always be taken into consideration when designing an OPC client.

"Synchronous" reading (**"Step: 11"**) of subscription data is effected via the "opcValueResults = Me.m_OpcServer.Subscriptions[0].Read()" function re-

quest and the transfer of the items to be read "Me.m_OpcServer.Subscriptions[0].Items" as parameters.

In this example the entire group is read and the result is forwarded to the "OnReadComplete" function for processing.

The "OnReadComplete" function is described in [chapter "Evaluating data" on page 213](#).

Asynchronous reading

With "asynchronous" access the task is transferred to the server and the function call returns immediately. The server processes the order and reports the order processing event back to the client via a pre-defined "delegate" ("OnReadComplete" event handler). This means that the client can **accept and process further actions** via the server during order processing.

"Asynchronous" reading (**"Step: 12"**) is effected, as for "synchronous" reading, via the "opcValueResults = Me.m_OpcServer.Subscriptions[0].Read()" function call.

The call differs only in the number of parameters to be transferred. The transfer parameters and their description are listed in the following table.

Transfer parameters	Description
Opc.Da.Item[]	Item collection of all items to be read. As with synchronous reading.
Object	ID transferred from the client which is given to the "delegate" when processing the results (optionally "null").
ReadCompleteEventHandler	"Delegate" for processing the results of the enquiry. The "delegate" is to be transferred to the function call using the instruction "new ReadCompleteEventHandler(OnReadComplete)".
IRequest	Handle to interrupt the request via the "Subscription.Cancel()" function call (optionally "null").

Fig.6-51: Transfer parameter for asynchronous reading

```

// check command type - sync or async
if (this.RadioButton_Sync.Checked == true)
{
    // Step: 11
    //
    // synchronous read of data
    Opc.Da.ItemValueResult[] opcValueResults = this.m_OpcServer.Subscriptions[0].Read(
        this.m_OpcServer.Subscriptions[0].Items);
    this.OnReadComplete(null, opcValueResults);
}
else if (this.RadioButton_Async.Checked == true)
{
    // Step: 12
    //
    // asynchronous read of data
    Opc.IRequest opcRequest;
    this.m_OpcServer.Subscriptions[0].Read ( this.m_OpcServer.Subscriptions[0].Items,
        null, new Opc.Da.ReadCompleteEventHandler ( OnReadComplete ), out opcRequest );
}

```

Fig.6-52: Synchronous and asynchronous reading of control data

Evaluating data

The "OnReadComplete" function for evaluating data returned from the OPC server is called following the "synchronous" reading from the OPC client and

Client Development

as an "event handler" ("Delegate") by the OPC server when providing the data following the "asynchronous" request.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread, it must be verified whether the "OnReadComplete" function call is running in the same thread as the thread of the form. This is verified in **"Step: 13.1"**. If the function has not been called in the thread of the form the request is synchronized by the "this.BeginInvoke()" function.

In **"Step: 13.2"** the results are evaluated and displayed in the list.

```
try
{
    // Step: 13.1
    //
    // check if it has to be synchronized with the thread of the form
    if ( this.InvokeRequired == true )
    {
        this.BeginInvoke ( new Opc.Da.ReadCompleteEventHandler ( OnReadComplete ),
            new Object[] { clientHandle, opcValueResults } );
        return;
    }

    // stop updating
    this.ListView_ItemData.BeginUpdate();

    // Step: 13.2
    //
    // update item values
    ListViewItem listViewItem;
    foreach (Opc.Da.ItemValueResult opcValueResult in opcValueResults)
    {
        listViewItem = this.ListView_ItemData.FindItemWithText ( opcValueResult.ItemName );

        if ( ( listViewItem != null ) && ( opcValueResult.Value != null ) )
        {
            listViewItem.SubItems[1].Text = this.GetValueAsString ( opcValueResult );
            listViewItem.SubItems[2].Text = opcValueResult.Quality.ToString ( );
        }
    }
}
catch ( Exception except )
```

Fig.6-53: Evaluating data - reading

Writing Control Data

General

This section describes write access to control data.

A distinction is made between

- "synchronous" and
- "asynchronous" access.

Control data is written in the "Button_Write_Click" event handler of the **"Write"** button.

The relevant code segments in the OPC client example project are:

- **"Step: 14"** - synchronous writing
- **"Step: 15"** - asynchronous writing
- **"Step: 16"** - evaluate data

Synchronous writing

With "synchronous" access the client waits until the server has completed the write task. The data is written in the "cache" and in the "device" of the OPC server if the subscription and the item are active. The client does not accept **any further operating actions** during this period, i.e. the "synchronous" call is blocking.

"Synchronous" writing (**"Step: 14"**) of subscription data is effected via the "opcResults = Me.m_OpcServer.Subscriptions[0].Write()" function call and the transfer of the "Opc.Da.ItemValue" ("opcValues") to be written as parameters.

In this example only one item from the entire subscription is written and the write task result is forwarded to the "OnWriteComplete" function for processing.

The "OnWriteComplete" function is described in [chapter "Evaluating data" on page 215](#).

Asynchronous writing

With "asynchronous" access the task is transferred to the server and the function call returns immediately. The server then processes the task and reports the order processing result back to the client via a pre-defined "delegate" ("OnWriteComplete" event handler). This means that the client can **accept and process further actions** via the server during task processing.

The "asynchronous" writing (**"Step: 15"**) is effected, as for "synchronous" writing via the "opcResults = this.m_OpcServer.Subscriptions[0].Write()" function call.

The call differs only in the number of parameters to be transferred. The transfer parameters and their description are listed in the following table.

Transfer parameters	Description
Opc.Da.ItemValue()	Item collection of all items to be written with values. As with synchronous writing.
Object	ID transferred from the client which is given to the "delegate" when processing the results (optionally "null").
WriteCompleteEventHandler	"Delegate" for processing the results of the task. The "delegate" is to be transferred to the function call using the instruction "new WriteCompleteEventHandler(OnWriteComplete)".
IRequest	Handle to interrupt the order via the "Subscription.Cancel()" function call (optionally "null").

Fig.6-54: Transfer parameters for asynchronous writing

```

// check command type - sync or async
if ( this.RadioButton_Sync.Checked == true )
{
    // Step: 14
    //
    // synchronous write
    Opc.IdentifiedResult[] opcIdentResults =
        this.m_OpcServer.Subscriptions[0].Write ( opcItemValues );
    this.OnWriteComplete(null, opcIdentResults);
}
else if ( this.RadioButton_Async.Checked == true )
{
    // Step: 15
    //
    // asynchronous write
    Opc.IRequest opcRequest;
    this.m_OpcServer.Subscriptions[0].Write ( opcItemValues, null,
        new Opc.Da.WriteCompleteEventHandler ( OnWriteComplete ), out opcRequest );
}

```

Fig.6-55: Synchronous and asynchronous writing of control data

Evaluating data

The "OnWriteComplete" function serves to evaluate the results delivered back from the OPC server. It is called by the OPC client after "synchronous"

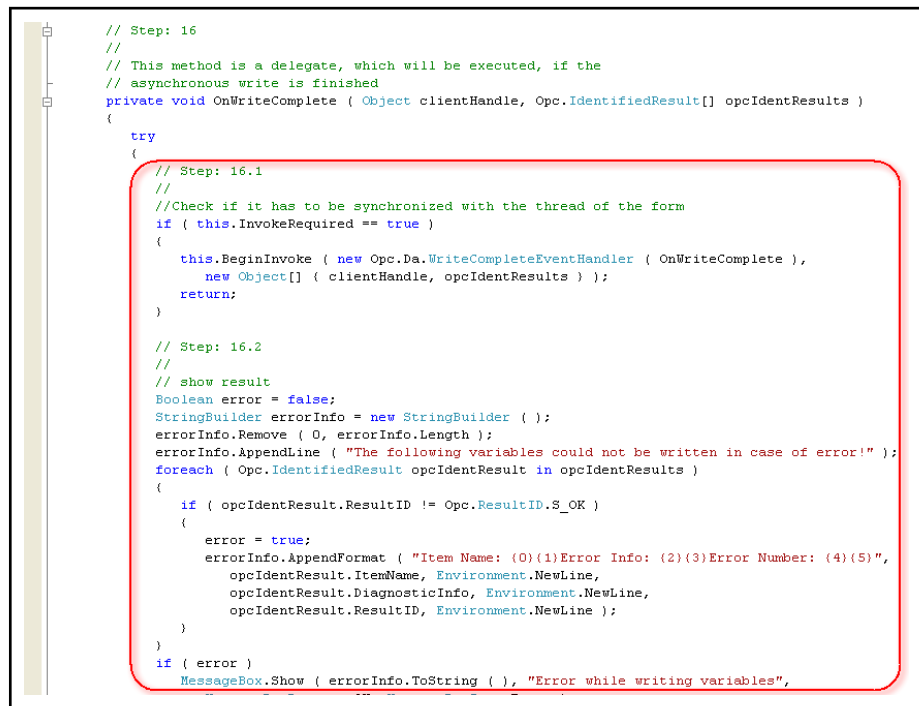
Client Development

writing and as an "event handler" ("delegate") from the OPC server, with the completion of task processing following the "asynchronous" command.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread, it must be verified whether the "OnWriteComplete" function call is running in the same thread as the thread of the form. This is verified in **"Step: 16.1"**.

If the function is not called in the thread of the form, the call is synchronized by the "this.BeginInvoke()" function.

In **"Step: 16.2"** the results of the write task can be evaluated. This evaluation can be performed in a "log file" or e.g. in a "StatusStrip" control element of the application. The example client does not contain an evaluation of the results of the write task.



```
// Step: 16
//
// This method is a delegate, which will be executed, if the
// asynchronous write is finished
private void OnWriteComplete ( Object clientHandle, Opc.IdentifiedResult[] opcIdentResults )
{
    try
    {
        // Step: 16.1
        //
        // Check if it has to be synchronized with the thread of the form
        if ( this.InvokeRequired == true )
        {
            this.BeginInvoke ( new Opc.Da.WriteCompleteEventHandler ( OnWriteComplete ),
                               new Object[] { clientHandle, opcIdentResults } );
            return;
        }

        // Step: 16.2
        //
        // show result
        Boolean error = false;
        StringBuilder errorInfo = new StringBuilder ( );
        errorInfo.Remove ( 0, errorInfo.Length );
        errorInfo.AppendLine ( "The following variables could not be written in case of error!" );
        foreach ( Opc.IdentifiedResult opcIdentResult in opcIdentResults )
        {
            if ( opcIdentResult.ResultID != Opc.ResultID.S_OK )
            {
                error = true;
                errorInfo.AppendFormat ( "Item Name: {0}{1}Error Info: {2}{3}Error Number: {4}{5}",
                                         opcIdentResult.ItemName, Environment.NewLine,
                                         opcIdentResult.DiagnosticInfo, Environment.NewLine,
                                         opcIdentResult.ResultID, Environment.NewLine );
            }
        }
        if ( error )
            MessageBox.Show ( errorInfo.ToString ( ), "Error while writing variables",
                               MessageBoxButtons.OK, MessageBoxIcon.Error );
    }
    catch { }
}
```

Fig.6-56: Evaluating data - writing

Cyclic Reading of Control Data

General

This section describes cyclic reading of control data.

The prerequisite for cyclic reading of values is that

- the subscription and
- the item are active and that
- an event handler is logged on to the subscription.

Activation and deactivation of cyclic reading are controlled via the "RadioButton_CheckedChanged" of the "Radio Buttons" "Synchronous", "Asynchronous" and "Cyclic" radio buttons.

The relevant code segments in the OPC client example project are:

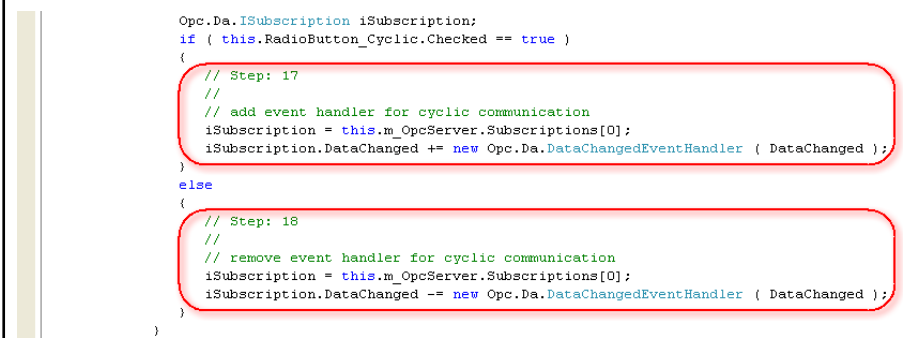
- **"Step: 17"** - activate cyclic reading
- **"Step: 18"** - deactivate cyclic reading
- **"Step: 19"** - evaluate data

Activating cyclic reading

As the prerequisites of the active subscription and active items have already been met, the "DataChanged" event handler only has to be added to the "iSubscription.DataChanged" subscription list (**Step: 17**). Adding the event handler fulfills all prerequisites and the client receives the "DataChanged" events.

Deactivating cyclic reading

To deactivate cyclic reading, the "DataChanged" event handler is removed again from the "iSubscription.DataChanged" subscription list. Once it has been removed, no further value changes can be reported to the client.



```

Opc.Da.ISubscription iSubscription;
if ( this.RadioButton_Cyclic.Checked == true )
{
    // Step: 17
    // add event handler for cyclic communication
    iSubscription = this.m_OpcServer.Subscriptions[0];
    iSubscription.DataChanged += new Opc.Da.DataChangedEventHandler ( DataChanged );
}
else
{
    // Step: 18
    // remove event handler for cyclic communication
    iSubscription = this.m_OpcServer.Subscriptions[0];
    iSubscription.DataChanged -= new Opc.Da.DataChangedEventHandler ( DataChanged );
}
    
```

Fig. 6-57: Adding and removing event handling

Evaluating data

The "DataChanged" function is requested cyclically by the OPC server at the time intervals specified in the "UpdateRate" element of the "Opc.Da.SubscriptionState" class.

All items are transferred to the "DataChanged" function with the corresponding values if

- the value has changed or
- the maximum "age" of the value was exceeded.

This means it is dependent on the configuration of the "MaxAge" and "MaxAgeSpecified" elements of the "Opc.Da.Item" class.

As control elements are not "thread-safe", i.e. these objects can only be accessed by one thread, it has to be verified whether the "DataChanged" function thread is running in the same thread as the thread of the form. This is verified in **Step: 19.1**.

If the function is not called in the thread of the form, the call is synchronized by the "this.BeginInvoke()" function.

In **Step: 19.2** the results are evaluated and displayed in the list view as with synchronous and asynchronous reading.

Client Development

```

// Step: 19
//
// event handler to receive callbacks from opc server during cyclic
// read of data
private void DataChanged(Object subscriptionHandle, Object requestHandle,
    Opc.Da.ItemValueResult[] opcItemValueResults)
{
    try
    {
        // Step: 19.1
        //
        // check if it has to be synchronized with the form thread
        if ( this.InvokeRequired == true )
        {
            this.BeginInvoke ( new Opc.Da.DataChangedEventHandler ( DataChanged ),
                new Object[] { subscriptionHandle, requestHandle, opcItemValueResults } );
            return;
        }

        // stop updating
        this.ListView_ItemData.BeginUpdate ( );

        // Step: 19.2
        //
        // update item values
        this.OnReadComplete ( null, opcItemValueResults );
    }
    catch ( Exception except )
    {
    }
}

```

Fig.6-58: Evaluating data - cyclic reading

Error Handling

Only a minimum of error handling was implemented with the example OPC client. Errors which arise with an OPC client used on production machines should be evaluated and recorded in more detail.

The OPC client is therefore restricted to verifying variables and simple exception handling in the form of

- "If" queries and
- exception handling with "try...catch..." instructions

"If" queries

"If" enquiries check the parameters required to perform the function. If the specified definitions cannot be fulfilled the function will be interrupted at this point.

Exception handling with "try...catch..." instructions

As the majority of components report an error to the caller during execution of a function by "throwing" an exception, at least one "Try...Catch..." statement should be included in every function/method. In the example OPC client only one "try...catch..." instruction was programmed due to the scope of the individual functions.

7 Service and Support

The Bosch Rexroth service helpdesk at our headquarters in Lohr, Germany, and our worldwide service provide 24/7 support and assistance.

	Helpdesk	Worldwide service hot-line
Phone	+49 (0) 9352 40 50 60	Outside Germany, please contact your sales/service office first.
Fax	+49 (0) 9352 40 49 41	
E-mail	service.svc@boschrexroth.de	For hotline numbers, refer to the sales office addresses on the internet.
Internet	For additional information on service, maintenance (e.g. delivery addresses) and training, please go to http://www.boschrexroth.com	

Required information

Please provide the following information:

- Detailed description of malfunction and circumstances leading to the malfunction
- Type plate name of the affected products, in particular type codes and serial numbers
- Phone and fax numbers as well as your e-mail address

Index

A

Abbreviations.....	12
About this documentation.....	7
Information representation	11
Validity of the documentation	7
Activating device configuration.....	89
Adding devices.....	86
Adding Windows form and control element.....	177

B

BRWS driver.....	92
Application area	92
Configuration	92
Documentation	100
Driver sheets	94
Error messages	99
Message information	99
PLC variable and parameter support	96
Properties	92
Protocol structure	96
Volume of data	96

C

Changing device OPC alias.....	88
Communications sample via OPC UA	
Subscribe and write	124
Compact industrial PC VPB 40.3 with operating device DP 08.3/16.3/40.3	
Field of application	42
Compact industrial PC VPB 40.3 with operating device VDP 08.3/16.3/40.3	
Accessories	44
Brief description	41
Documentation	45
Models	43
Technical data	42
Wear parts	42
Compact industrial PC VPP 16.3/40.3/60.3	
Accessories	47
Brief description	45
Documentation	48
Field of application	46
Models	47
Technical data	46
Wear parts	47
Configuration of the IndraLogic OPC.DA server	
Configuration server	101
Configuring IndraLogic OPC.DA server	
File location Windows XP	109
Item syntax	111
OPCServer.ini	105
Configuring the IndraLogic OPC.DA server.....	101
File location	109
File location Windows CE	109

C

Configuring the IndraLogic OPC.DA Server	
Identifying the communication address	101
Creating device configuration.....	86

D

Displaying device properties.....	87
Drive system.....	15

E

Electric drive system.....	15
Error messages Bosch Rexroth WinStudio driver.....	99
Exiting the application.....	89

F

File location - OPC.DA server.....	109
Windows CE	109
Windows XP	109

H

High-end industrial PC.....	41
-----------------------------	----

I

Identifying the communication address.....	101
Implementing OPC client.....	179
Important instruction for use.....	13
Improper use.....	14
IndraControl VCH 08 - Compact hand-held terminal for mobile use	
Brief description	31
IndraControl VCH 08 - Compact hand-held terminal for mobile use	
Field of application	32
IndraControl VCH 08 - Compact hand-held terminal for mobile use	
Accessories	33
Documentation	33
Models	32
Technical data	32
IndraControl VEH 30.2	
Brief description	36, 38
Field of application	37
Models	38
Technical data	37
Wear parts	38
IndraControl VEP 30.3 / 40.3 / 50.3 / VEP 30.4 / 40.4 / 50.4 - Embedded PC-based operating terminals	
Accessories	36
Brief description	33
Documentation	36

Index

I

...IndraControl VEP 30.3 / 40.3 / 50.3 / VEP 30.4 / 40.4 / 50.4 – Embedded PC-based operating terminals	
Models	34
Technical data	34
Wear parts	34
IndraControl VEP 330.3 / 40.3 / 50.3 / VEP 30.4 / 40.4 / 50.4 – Embedded PC-based operating terminals	
Brief description	33
IndraLogic Items.....	91
Information representation	
Names and abbreviations	12
Safety instructions	11
Symbols used	11
Intended use.....	13
Scope of application	13
Intended Use	
State of the art	13
Introducing the OPC Unified Architecture	
General information	112
OPC UA information model	117
Service-oriented architecture	113
Specification overview	112
Item syntax.....	111
lwSCP Items.....	89
lwSCP Server	89

M

ModbusTCP.....	162
Configuration with Pro-face GP-Pro EX 2.2	167
Configuration with Siemens WinCC-flexible 2007	163
General	162
Modbus TCP.....	127
Introduction and overview	127

N

Non-Intended Use	
Consequences, Exclusion of Liability	13

O

OPC.DA Server.....	160
Configuration	160
Transfer configuration on a visualization device VEPxx.4 with WindowsCE	161
OPC.lwSCP Items.....	89
OPC.lwSCP Server	89
OPC Client Visual Basic .NET.....	175
Exception handling with "Try...Catch..." instructions	197
If queries	197
Visual Basic project "OPC Client "	175
Visual C# Project "OPC Client"	198

O

OPC Client Visual C#.NET	
If queries	218
OPC Client Visual C# .NET.....	197
Activating cyclic reading	217
Adding and removing OPC items	209
Adding items - AddItems	209
Adding References	198
Adding Windows form and control elements	199
Asynchronous reading	213
Asynchronous writing	215
Canceling a subscription - CancelSubscription	207
Creating and canceling OPC server subscription	206
Creating and deleting OPC server instance	203
Creating a subscription - CreateSubscription	206
Creating OPC server instance - Connect	204
Cyclic reading of control data	216
Deactivating cyclic reading	217
Declaring variables	202
Deleting OPC server instance - Disconnect	204
Displaying subscription information	208
Error handling	218
Evaluating data	213, 215, 217
Exception handling with "try...catch..." instructions	218
Implementing OPC client	201
IndraLogic OPC server	210
Listing control items	208
Listing OPC server	203
New project	198
Opc.Da.BrowseFilter	208
Opc.ItemIdentifier	208
OPC.lwSCP server	210
Reading control data	212
Reading control items	208
Reading server information	205
Removing items - RemoveItems	211
Synchronous reading	212
Synchronous writing	214
Writing control data	214
OPCServer.ini.....	105
OPC UA.....	111
Bosch Rexroth OPC UA server	117
Introducing the OPC Unified Architecture ..	112
OPC UA communication sample	119
Overview	111
OPC UA Communication sample	
Browsing	123
Establishing a connection	121
Introduction	119
Requirements	120

O

...OPC UA Communication sample UaExpert	120
--	-----

P

PELV.....	20
PROFIBUS DP.....	30
Protective extra-low voltage.....	20

R

Removing devices.....	88
-----------------------	----

S

Safety instructions for electric drives and controls.....	15
Saving device configuration.....	88
Saving device configuration as.....	88
SCP OPC configurator	
General	85
Operation	86
SCP-OPC Configurator.....	85
Activating device configuration	89
Adding devices	86
Changing device OPC alias	88
Creating device configuration	86
Displaying device properties	87
Exiting the application	89
Opening device configuration	86
Removing devices	88
Saving device configuration	88
Saving device configuration as	88
Screen size.....	65
sercos items.....	91
Small Control Panel VCP	
Accessories	30
Brief description	27
Small operator panel VCP	
Documentation	31
Models	29
Technical data	29
Standard industrial PCs.....	48
Standard industrial PC VSB 40.3 with operating device VDP 08.3/16.3/40.3	
Accessories	40
Brief description	38
Documentation	41
Field of application	39
Models	40
Technical data	39
Wear parts	39
Standard industrial PC VSP 16.3/40.3	
Accessories	50
Brief description	48
Documentation	50
Field of application	49
Models	50

S

...Standard industrial PC VSP 16.3/40.3	
Technical data	49
Wear parts	49
Support	
see Service-Hotline	219

V

VDP 08.3.....	41
VDP 16.3.....	41
VDP 40.3.....	41
VI-Composer	
Communication protocol BRC-symbolic	54
Configurable small operating terminals	53
Connection settings	54
Documentation	55
Field of application	53
VI-Logic	
Documentation	53
Field of application	51
Properties of the integrated visualization	51
Protocol	53
Supported visualization devices	53
Visualization objects	52
Visual Basic project"OPC client"	
IndraLogic OPC server	188
OPC.lwSCP Server	188
Visual Basic project"OPC Client"	
Adding Windows form and control element	177
Visual Basic project "OPC client"	
Activating cyclic reading	195
Adding items - AddItems	187
Adding references	176
Asynchronous reading	191
Asynchronous writing	193
Creating OPC server instance - Connect ...	182
Cyclic reading of control data	195
Deactivating cyclic reading	195
Declaring variables	180
Deleting OPC server instance - Discon-	
nect	182
Displaying subscription information	186
Error handling	197
Evaluating data	192, 194, 196
Listing control items	186
Listing OPC server	181
New project	175
Opc.Da.BrowseFilter	186
Opc.ItemIdentifier	186
Reading control data	190
Reading control items	186
Reading server information	183
Removing items - RemoveItems	189
Synchronous reading	190
Synchronous writing	193
Writing control data	193
Visual Basic project "OPC Client".....	175

Index

V

...Visual Basic project "OPC Client"	
Canceling an OPC subscription - Delete-Subscription	185
Creating an OPC subscription - Create-Subscription	184
Implementing OPC client	179
VPB 40.....	41

W

WinStudio.....	55
Activating a WinStudio license	82
Alarms	71
Consequences of a missing license or a license of a too low class	82
Creating and editing WinStudio screens	65
Creating a WinStudio application	83
Cross reference search	79
Database spy	78
Export/import	80
Further information	84
General WinStudio settings and functions ...	57
Graphic Script	75
Linked symbols	68
LogWin	79
Mathematic sheets	70
Multilingualism	83
Object properties and variable selection	67
ODBC	74
Overview	55, 83
Recipes	72
Reports	72

W

...WinStudio	
Screen properties	65
Screen scripts	76
Scripts	74
Sequences	73
Start application	78
Start Script	75
Transmitting the application and activation on the target device	84
Trends	73
User management	83
User scripts	76
User Symbols	69
Variable import wizard	80
Verify application	77
Web functionality	84
WinStudio applications	83
WinStudio elements in the Project Explorer .	56
WinStudio library	68
WinStudio license overview	81
WinStudio licensing	81
WinStudio screens	65
WinStudio tasks	70
WinStudio tools	77
WinStudio variables	67
WinStudio with IndraLogic.....	133
Creating the IndraWorks project	133
Creating the WinStudio visualization	148
Integrating the WinStudio visualization in IndraWorks	140
Starting the interface	156
Technical requirements	133

Notes

Bosch Rexroth AG
Electric Drives and Controls
P.O. Box 13 57
97803 Lohr, Germany
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Germany
Tel. +49 (0)93 52-40-0
Fax +49 (0)93 52-48 85
www.boschrexroth.com/electrics



R911334184